JPRS L/10064

22 October 1981

USSR Report

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

(FOUO 23/81)

FBIS FOREIGN BROADCAST INFORMATION SERVICE

NOTE

JPRS publications contain information primarily from foreign newspapers, periodicals and books, but also from news agency transmissions and broadcasts. Materials from foreign-language sources are translated; those from English-language sources are transcribed or reprinted, with the original phrasing and other characteristics retained.

Headlines, editorial reports, and material enclosed in brackets [] are supplied by JPRS. Processing indicators such as [Text] or [Excerpt] in the first line of each item, or following the last line of a brief, indicate how the original information was processed. Where no processing indicator is given, the information was summarized or extracted.

Unfamiliar names rendered phonetically or transliterated are enclosed in parentheses. Words or names preceded by a question mark and enclosed in parentheses were not clear in the original but have been supplied as appropriate in context. Other unattributed parenthetical notes within the body of an item originate with the source. Times within items are as given by source.

The contents of this publication in no way represent the policies, views or attitudes of the U.S. Government.

APPROVED FOR RELEASE: 2007/02/09: CIA-RDP82-00850R000400060041-6

JPRS L/10064

22 October 1981

USSR REPORT

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

(FOUO 23/81)

CONTENTS

HARDWARE

DIGITAL CORRELATOR

Moscow OPISANIYE IZOBRETENIYA in Russian No 739544, 1980

[Author certificate by S. N. Britin, V. P. Ipatov, Yu. A. Kolomenskiy and B. I. Korniyevskiy]

[Text] The invention relates to digital computer technology. It can be used for determining the correlation functions of various processes. It can also be used in systems and devices based on correlation methods of signal processing, in particular, radar and radio navigation systems.

A previously known digital correlator contains four dynamic memory units, two multiplication units, two accumulators, and a register (1).

This single-channel device is relatively complex and has insufficient speed. The presence of multipliers in the correlators is the usual reason for their relatively slow speed. The use of high-speed memory devices as multipliers is not always possible, because the number of quantized signal levels can reach a significant size and because this then requires an excessively large memory.

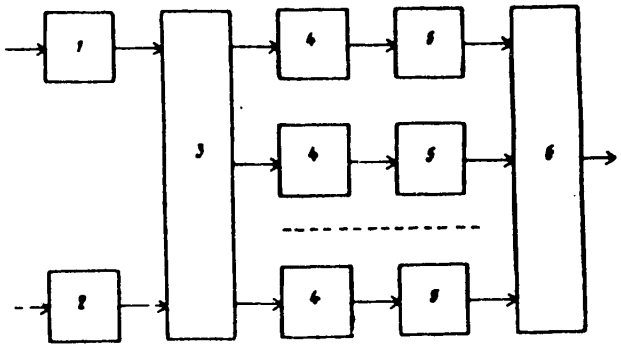
Also already known is a digital correlator containing analog-to-digital converters for the input and reference signals. Their outputs are attached via the multiplier to the adder, which serves as integrator. The signal on adder output determines the size of the correlation of the reference and input signals (2).

The correlator's low speed is a drawback. This is caused by the following. It is necessary to increase the number of quantization levels of the input signal in order to increase the correlator's accuracy and to eliminate the suppression of weak signals. In this case, the multiplication time, which is proportional to the number of digits in the multipliers, increases. This in turn increases the minimally permitted time quantization sample of the input processes; that is, it lowers the correlator's speed.

The goal of this invention is an increase in digital correlator speed.

This goal is achieved as follows. The correlator contains first and second analog-to-digital converters (their inputs are the inputs of the correlator), multiplication units, accumulators and equalizer. Included in the correlator

is a switch. The number of the switch's outputs equals the number of quantization levels of the reference signal. The switch's inputs are connected to the outputs of the corresponding analog-to-digital converters, and the outputs are connected to the inputs of the corresponding accumulators. The output of each accumulator is connected to the input of the multiplication unit. The output of each multiplication unit is attached to the corresponding equalizer input. This allows during input signal operation the execution of solely the summing of its quantized selected values. Each value corresponds to a definite quantization level of the reference signal. The slower multiplication operation, which limits the operating speed, is executed following the completion of the input signal operation.



The diagram shows the structural organization of the digital correlator.

The correlator consists of analog-to-digital converters (1 and 2) of the input and reference signals, whose outputs are connected to the switch (3). The number of switch outputs equals the number of quantization levels of the reference signal. Each output is attached to the corresponding adder (4), which is linked to the corresponding multiplication unit (5). All outputs from number 5 units are attached to the equalizer's inputs (6).

The digital correlator operates in the following manner.

The input signal arrives at the analog-to-digital converter (1), where sampling and level quantization occur. From the output of the analog-to-digital converter (1) the signal, which is an n-digit number, goes to the switch (3). The switch is controlled by signals arriving from the analog-to-digital converter (2) of the reference signal. At each moment of the sampling, the selected quantized value of the input signal (going through switch (3)) arrives at one of the adders (4), which corresponds to a definite quantized level of the reference signal. Following completion of the input signal, amounts stored in the adders (4) are multiplied in the multiplication units (5) by coefficients which

equal the corresonding values of the quantization levels of the reference signals. The resulting products undergo final addition at the equalizer (6). The result determines the desired correlation value.

Example: Signals from the outputs of the analog-to-digital converters (1 and 2) are represented by 10-digit numbers.

Let the execution time for one addition operation equal t_c; then, for ordinary multiplication devices, the execution time for one multiplication operation is approximately $t_m = 20t_c$.

The quantized signal sample in a known correlator (prototype) must be selected according to the condition $t_k \geq t_c + t_m = 20t_c$, while in the described correlator the condition is $t_k \geq t_c$.

The speed of the digital correlator described here is approximately 20 times greater than the speed of the prototype.

In practice there are often cases when a digital multidigit (for example, 10 digit) correlator reference signal accepts only 2 values. Then an increase in equipment, in comparison with the prototype, is not large. Specifically, a switch, an adder, a multiplier and an output adder are added. When the correlator's elements are implemented by integrated circuits, the increase in equipment volume is entirely acceptable (if one considers that correlator speed will then increase 20 times).

Invention Formula

The digital correlator contains first and second analog-to-digital converters (whose inputs are the inputs of the correlator), multiplication units, accumulators and an equalizer. The digital correlator is unique in that, in order to increase correlator speed, it contains a switch such that: the number of the switch's outputs equals the number of reference signal quantization levels; the correlator inputs are connected with the outputs of the corresponding analog-to-digital converters; and the outputs are connected to the inputs of the corresponding accumulators. The output of each accumulator is connected to the multiplication unit input, each of whose output is attached to the corresponding equalizer input.

Information sources considered during evaluation

1. USSR Author Certificate No 468247, class G 06 F 15/34, 1974.

2. Gribanov, Yu. I. et al. "Automatic Digital Correlators," Moscow, Izdatel'stvo "ENERGIYA," 1971, 150 pp (prototype)

CSO: 8144/1846-P

UDC 519.47

PRINCIPLES OF ORGANIZATION OF PARALLEL COMPUTATIONS. STRUCTURE AND REALIZATION OF COMPUTER SYSTEMS: THE M-10

Kiev KIBERNETIKA in Russian No 2, Mar-Apr 81 (manuscript received 9 Dec 79) pp 68-74

[Article by Professor Mikhail Aleksandrovich Kartsev, doctor of technical sciences, Moscow]

[Text] We will apply the term "computer systems" to computer hardware intended for the performance of parallel computations [1].

By now at least two situations, different in principle, in which computer systems are used, have been formulated. The requirements for the structure of the computer systems and the parameters on the basis of which their possibilities must be evaluated are correspondingly different.

The first typical situation is the use of a computer system in a shared multicomputer center or automated system for the control of production, of the educational process in a VUZ, etc. Characteristic of this situation is the passage through the system of a large flow of tasks independent or almost independent of one another; each of them could be solved with an ordinary computer. The sense of using a computer system instead of an aggregate of separate machines in a shared multicomputer center consists in the possibility of more effective use of collectivized resources (professorial time, internal memory, peripherals, etc), in an economy of the volume of information stored in external memories (for example, general purpose programs can be stored in external memories in a single copy and copied as needed into the internal memory several times in the interests of different tasks), and also in increased reliability of the shared multicomputer center. When a computer center is used to solve ASU tasks an additional and, possibly, a deciding argument in favor of use of the computer center is the need to refer different ASU tasks to a common data bank.

The second typical situation is the use of a computer system to control a complex technological process in real time or to solve large-scale scientific or engineering problems. It is characteristic here that all the resources of the computer system are made available if possible to a single priority task (if it can use those resources); that task is so labor-intensive that no combination of separate computers of the ordinary type can assure the obtaining of a solution in an acceptable time.

Also possible are situations in which the purpose of the creation of the computer system is the achievement of greater reliability than with an ordinary machine or

simplification of data transmission channels and the obtaining of a gain in relation to some other factors, but these situations will not be discussed. The main attention has been concentrated on a typical situation in which it is necessary to obtain from a computer system a high rate of solution of a single separately taken task. If the system is not strictly specialized, then in another time segment it will be used to solve another large-scale task, but in that case a high rate of solution of the given task will again be required, and not a high rate of passage of a flow of many relatively small-scale tasks.

When it is a matter of some specific computer system, usually reported among its parameters is the nominal system speed, obtained as the sum of the speeds of its components. However, the nominal speed characterizes not so much the possibilities of the system as the expenditures on its creation. To characterize the possibilities of a system in the first typical situation it is necessary to know its real system speed, that is, the speed of the system during the passage through it of a large flow of relatively small-scale independent or almost independent tasks, and in the second, the real user speed, that is, the speed of the system during the solution of a single separately taken large-scale task. The ratio of the real speed of a computer system (system or user) to the nominal is called the system or user efficiency of the computer system.

Obtaining high user productivity in the presence of adequate efficiency of the computer system is a very complex problem. Achievement of that goal requires a definite correspondence between the structure of the computer system and the properties of the task to be solved. A given computer system cannot be identically efficient in solving tasks of different classes.

Three characteristics can be distinguished which are to a different degree intrinsic to different tasks and can be used for the organization of parallel computations during their solution (three types of parallelism).

1. Parallelism of independent branches of a task. Separate large sections of a task can be solved independently of one another, only interchanging results with one another from time to time.

2. Parallelism of related operations. The result of performance of some operation is not used without fail as the input operand in the operation following it, and perhaps is not used also in several subsequent operations, as a result of which those related operations can be performed simultaneously.

3. Parallelism of a set of objects and natural parallelism. It is characteristic of tasks in processing information on a large number of identical or almost identical objects, in other words, of tasks dealing with data files. This is a particular case of a broader understanding of natural parallelism, when a task has been formulated from the very start in terms of the processing of multidimensional vectors or matrices, lattice functions or other objects similar to them [2, 3].

Different structures of computer systems correspond to the use of different types of parallelism.

Parallelism of independent branches is realized either by a multimachine complex or a multiprocessor system with separate control, where each processor contains

an autonomous control device and works in accordance with its own independent program (Fig 1) or a system of the conveyor (main-line) type containing just as many control devices in the conveyor, connected alternately (on a ring) to different stages of the conveyor, as there are stages in the conveyor. After a certain computer in the multimachine complex (or a certain processor of the multiprocessor system, or a certain control device in the main-line system) has completed the filling of its branch, there is a possibility of verifying somehow whether the other parallel branches have been completed, of organizing, if necessary, the expectation and exchange of information and of triggering the autonomous filling of following branches.

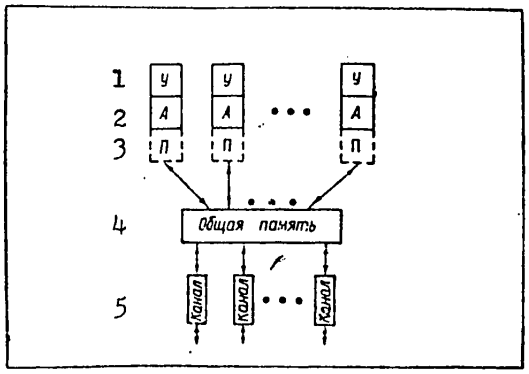


Figure 1. Example of construction of a multiprocessor system with separate control

1 -- control
2 -- arithmetic-logical unit
3 -- individual storage device of processors (can be absent)
4 -- common memory
5 -- channel

Unfortunately, the parameters proposed in scientific works for numerical estimation of the parallelism of independent branches intrinsic to a certain task relate not so much to the specific task being solved as to a specific program for solving it. But even detailed knowledge of those parameters far from always makes it possible to estimate the real user speed and efficiency of a computer system oriented toward parallelism of independent branches, in solving a given task. Therefore interest is presented by works in which those characteristics are determined by simulation of computational processes in solving real tasks. In particular, in one of those works [4] tasks of x-ray structure analysis, weather forecasting, etc, were investigated.

The results of all those works are more or less similar and are presented in averaged form on Fig 2, which shows the nominal speed $s_н$, the real user efficiency $s_p(n)$ and the user efficiency of the computer system E_n are shown as a function of n, the number of machines in a multimachine complex, of processors in a multiprocessor system with separate control or of independent control devices in a conveyor system.

It is evident from Fig 2 that, if the purpose of the creation of a computer system of the type under consideration is the achievement of high user speed in solving a broad circle of tasks, it is hardly advisable to specify n larger than 4-6. Sometimes systems of that kind are constructed with a larger number of machines (or

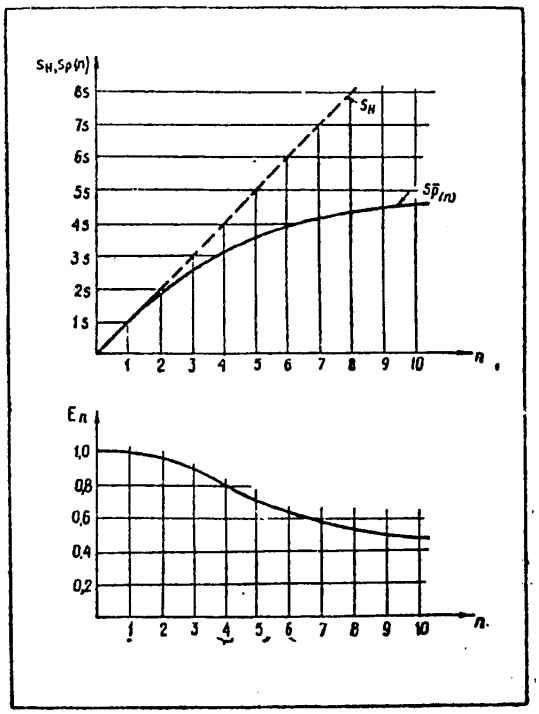


Figure 2. Speed and user efficiency of computer systems using parallelism of independent branches.

processors or control devices). In that case the purpose is mainly to achieve high system productivity. But if in the stream of tasks a relatively large-scale task appears, then a subsystem can be separated for it, one containing as many machines (processors or control devices) as that task can efficiently load. As a result, the system efficiency of the computer system obtained is high enough, but the user productivity in the solution of a single separately taken task, except special cases, cannot exceed by more than 3-4 times the productivity of an ordinary single-processor machine.

Parallelism of related operations is used differently in different developments. Its classical use is a multiprocessor system with general control and vectorial instruction, containing separate operational codes for different processors (Fig 3). It is possible to create conveyor (main-line) systems oriented toward this type of parallelism.

A numerical characteristic of the parallelism of related operations, intrinsic to a given task, is the indicator of connectedness of related operations α -- the probability that the result of some operation in a program will be used in the operation following it. The value of α depends not so much on the properties of the task as on the quality of the local optimization of the program, done in the process of translation into machine language or, in some cases, by apparatus directly in the process of interpreting the program. For many tasks of a

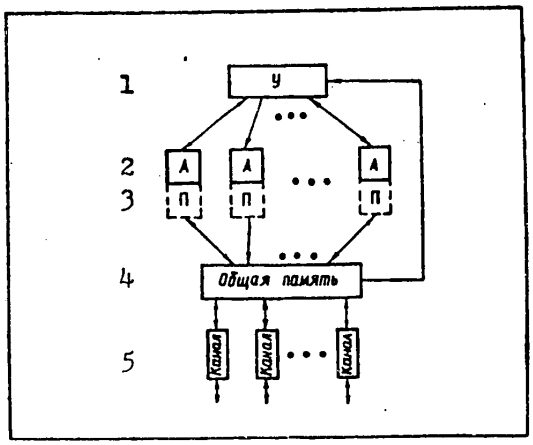


Figure 3. Example of construction of a multiprocessor system with common control, oriented toward parallelism of related operations.
1 to 5 -- as for Figure 1

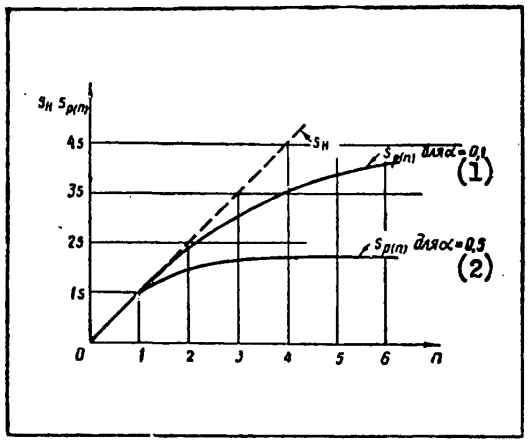


Figure 4. Speed of computer systems using parallelism of related operations.
1 -- for $\alpha = 0.1$ 2 -- for $\alpha = 0.5$

computational character the value of α obtained is more or less identical--from 0.1 when the program is optimized very carefully to about 0.5. It is evident from Fig 4 that, in a system oriented toward the use of parallelism of related operations, it is difficult to expect gains in user productivity as compared with an ordinary computer of more than 2.5-3 times; to obtain that, n (the number of processors in a multiprocessor system or stages of a conveyor) must be of the order of magnitude of 3-5.

Parallelism of a set of objects and natural parallelism are used in the structure of a multiprocessor system with general control (Fig 5), when all the processors simultaneously perform identical operations, but on different data (relating to different objects or different components of multidimensional vectors, etc). Analogous construction of a conveyor system also is possible. It is desirable to supplement that structure with a mask mechanism, a file sorting processor, and general arithmetic circuits for performing such operations as computation of the scalar derivative of vectors, finding the minimal or maximal element in a file, etc, for example.

The principal quantitative characteristic of parallelism of a set of objects or natural parallelism is the rank of the task r--the number of objects of the same kind in the set of objects being processed, or the dimensionality of multidimensional vectors, the number of points at which the values of functions are given, etc. It can readily be shown that the user efficiency of a multiprocessor system oriented toward parallelism of a set of objects or natural parallelism depends only on the ratio r/n, and the user efficiency of an analogous conveyor system--mainly on the ratio r/n and to a small degree on n. Here the value of n, as previously,

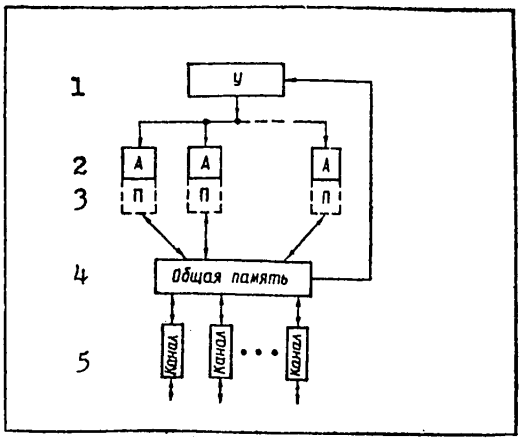


Figure 5. Example of construction of a multicompressor system with general control, oriented toward natural parallelism. 1 to 5 -- as for Figure 1

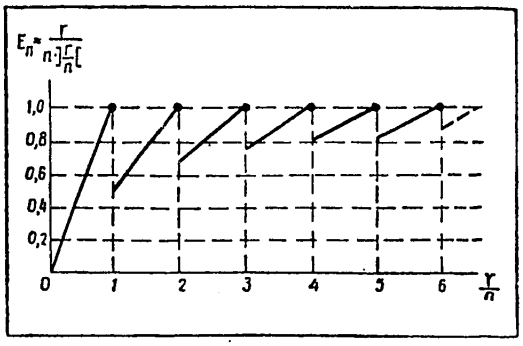


Figure 6. Efficiency of a multiprocessor system using natural parallelism.

is the number of processes in a multiprocessor system or the number of stages in a conveyor system. In a somewhat idealized form those dependences are presented for a multiprocessor system on Fig 6. It is evident from the figure that an r/n ratio larger than or equal to 4, for example, the user efficiencey will be at least 0.8 in any case. This means that at a rank of the tasks, for example, of not less than 100, 25 processors under general control can be efficiently used in the system, and in that case the gain in real user speed obtained is at least 20 times as compared with an ordinary computer, and at a rank of the tasks of at least 100, 250 processors can be used correspondingly (if other circumstances do not prevent that) and a gain in real user speed of at least 200 times can be obtained.

Similar results cannot be obtained during orientation toward parallelism of independent branches or of related operations. Of course, such a gain in real user speed is possible only for a certain class of tasks which have a high rank of natural parallelism or, in particular, parallelism of a set of objects. However, among large-scale tasks, for the solution of which computer systems with high user productivity are required, such tasks occupy a very important place.

Attention must also be turned toward a difference in the characteristics of programming when various types of parallelism and correspondingly different structures of computer systems are used for the organization of parallel computations.

When parallel independent branches and natural parallelism (parallelism of a set of objects) are used, new programming languages or supplements to existing ones are needed. Some means for description of parallel processes are available only in the PL/1 language.

It is necessary, however, to take into consideration the presence of a large fund of programs written in ALGOL-60, FORTRAN and other widespread languages, and the presence of a large number of programmers who are well acquainted with those languages and have experience in working with them.

Let a program be written in ALGOL-60 or FORTRAN without any sort of supplementary instructions; it is proposed to use parallelism of independent branches. In that case the translator must conduct an investigation on the object of the possibility of parallel execution of all sections of the program on a single level (blocks of ALGOL-60 or FORTRAN subroutines). It is not so complicated a matter to formulate the rules according to which the translator would determine the possibility of executing a pair of sections of the program in parallel. However, the work which the translator must do in that case is enormous for a large program, as an exhaustive search is required for all pairs of blocks of the ALGOL program or subroutines of the program in FORTRAN. Therefore when parallel independent branches are used it is necessary to require that the programmers mark the program in the source language by means of additional

If it is proposed to use natural parallelism or parallelism of a set of objects, and the program is written in ALGOL-60 or FORTRAN, for example, then the task of the translator consists in investigating the programming languages on the subject of the possibility of their parallel execution. The formal rules according to which the translator must act prove in that case to be of approximately the same complexity as the rules according to which the translator determines the possibility of parallel execution of two sections of the program during use of parallel independent branches. However, the work of the translator on the whole in that case is far simpler, since each program cycle is investigated separately (perhaps only together with the cycles built into it) and joint investigation of several cycles is not required.

The additional language resources which simplify this task could consist in the introduction into the language, together with such objects as files (which already exist in ALGOL-60 and FORTRAN), also of vectors, matrices and lattice functions (real, purpose and Boolean) and in the determination of operations on those objects.

Let us note that from the point of view of the programmer there is a substantial difference between those supplementary resources which are suitable for describing parallelism of independent branches and the means of description of natural parallelism. The introduction of supplementary instructions regarding the possibility of separating independent branches represents "excessive" work for the programmer, and the efficiency of the computational process which will be obtained during execution of the program will depend to a considerable degree on how well that work has been done (in other words, on the skill of the programmer). On the other hand, the introduction of such objects of language as files, vectors, matrices, functions and the determination of operations on them greatly simplifies the work of the programmer.

These tasks in the processing of large flows of information, which have to be solved in practice by means of software with high productivity, in many cases have to some degree all types of parallelism. To obtain high efficiency it is advantageous to construct combined systems [5]. For example, the M-10 computer (Fig 7) is oriented toward use of natural parallelism and parallelism of related operations [6].

The main processor part of the machine consists of two program-tuned lines of processors. Each line, depending on the operational code, represents either a

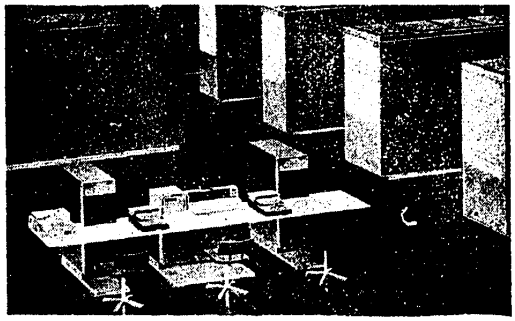


Figure 7. The M-10 computer.

16-digit figure of eight, a 32-digit tetrad or a pair of 64-digit processors which perform one and the same operation (on different data, of course). The other line of processors for a separate operational code contained in a vector-instruction can simultaneously perform the same or a different operation. Definite operational codes gather all the processors of the line in a single vectorial processor. For example, during performance of the operation "scalar product", in the line of processors in the course of a single cycle eight pairs of 16-digit or four pairs of 32-digit numbers are multiplied in pairs and the obtained products are summed with one another and with the sum accumulated in the preceding cycle.

Simultaneously with the performance of operations on numbers, in the main processor lines up to five lines of Boolean variables are produced in which each binary digit corresponds to a definite sign relating to operands participating in the operation or to the result of the operation. For example, during addition (of eight pairs of 16-digit or four pairs of 32-digit, or two pairs of 64-digit numbers), lines of Boolean variables are produced which contain 8, 4 or 2 binary digits respectively: ω --the sign of overfilling; e --the sign of equality of components with one another; m --the sign that the first component ψ in the given pair proved to be larger than the second; z --the sign of equality of the result to zero; x --the sign of obtaining a negative result.

These signs can be transmitted directly or through the memory into a special processor for processing lines of Boolean variables which acts simultaneously with the lines of main processors and is capable of executing the complete set of logical operations on Boolean variables. The lines of Boolean variables obtained directly during execution of the main arithmetic or logical operations on the numbers or as a result of the work of the indicated special processor, or read from the memory, can be used further to organize provisional transmissions of control or as "masks" to perform the basic arithmetic-logical operations. If in the operational code for the basic line of processors it is pointed out that the operation must be carried out under a mask, then in the line those processors are blocked for which the corresponding places of the mask contain zeros (and when the operation is performed under an "inverse" mask the processors are blocked for which the corresponding places of the mask contain units).

The mechanism of masks is an important tool in the efficient organization of parallel computations.

Still another specialized processor in the central part of the machine is intended for the execution of index operations.

For tasks which have sufficient natural parallelism the mean speed of the M-10 is more than 5 million operations per second.

The capacity of the internal memory of the machine is 5 Mbytes (1,310,720 words). In that case, to provide all the processors with operands, each reference to the memory is performed in a format of up to 64 bytes (up to 512 information positions) and the machine instructions are two-address.

The operating system of the machine assures users working in an interactive mode with time sharing access to the translators and means of debugging programs in algorithmic languages, reference on the logical level to peripherals and standard procedures, the use as finished modules for the charging of programs from the library of standard programs (linear algebra, approximation of functions, squaring, the integration of ordinary differential equations and equations in partial derivatives, etc).

External exchange of information is done through a multiplex channel with a total throughput capacity of about 7 Mbytes/second. The multiplex channel has 24 duplex subchannels, to each of which in turn up to six peripherals of the same kind can be connected. The basic complex of the machine includes as peripherals terminals constructed on the basis of a typewriter with punched-tape equipment, alphanumeric line-printers, punched card equipment and an engineering panel typewriter by means of which the apparatus journal of the machine is kept. Through additional coupling devices, terminals based on hatched displays with keyboards and a light pen can be connected to the channel, as well as magnetic tapes and disks and other YeS computer peripherals. The channel has its own buffer memory with a volume of 64 Kbytes, in which channel command words also are stored, but it can be addressed directly also to any cell of the internal memory of the machine.

In the machine provision has been made for circuits which permit combining up to seven M-10 computers in a single synchronous complex working from a common clock pulse generator. In each cycle the machine, working in the complex can issue to its output lines a data file of 64 bytes and receive a file of the same size from any other machine of the complex. In that case the addressing of the machines is virtual [5]. Special apparatus of the descriptors of connections efficiently replaces the virtual number of the machine, from which information is received, by a physical number. When necessary this must permit forming several subcomplexes within the synchronous complex. At the same time, as was supposed, together with natural parallelism and parallelism of related operations, parallelism of independent branches could be used in the complex, and increase of the system productivity could also be achieved. However, the possibility of organizing synchronous complexes of M-10 computers has not yet been realized anywhere in practice.

In the long term, obviously, it makes sense to become oriented toward simultaneous use of parallelism of independent branches (2 to 4 branches) and parallelism of a set of objects.

In that case the equipment of the system can be constructed so that programs for the solution of various tasks, even in a machine language, do not depend at all on the number of processors performing parallel processing of sets of objects, and look exactly the same as programs of an ordinary (one-processor) machine. In this way one achieves both complete program compatibility of systems different in their completeness of sets (number of processors) and high viability of the system (the possibility of continuing to work when some of the processors fail).

BIBLIOGRAPHY

1. Kartsev, M. A. "The Problem of Organizing Parallel Computations and the Structures of Computer Systems." Materialy Vsesoyuznoy konferentsii "Vychislitel'nyye sistemy, seti i tsentry kollektivnogo pol'zovaniya" (Materials of the All-Union Conference on "Computer Systems, Networks and Shared Multicomputer Centers") 4.1. Novosibirsk, 1980, pp 64-77.

2. Glushkov, V. M., Kapitonova, Yu. V., and Letichevskiy, A. A. "Theory of Data Structures and Synchronous Parallel Computations." KIBERNETIKA, 1976, No 6, pp 2-15.

3. Glivenko, Ye. V. "On One Principle of Parallel Computations." PROGRAMMIROVANIYE, 1977, No 1, pp 3-9.

4. Martin, D. F., and Estrig, G. "Experiments on Models of Computation and System." IEEE TRANSACTIONS ON ELECTRONIC COMPUTERS, EC-16, 1967, No 1, pp 67-79.

5. Kartsev, M. A. "Questions of the Construction of Multiprocessor Systems." VOPROSY RADIOELEKTRONIKI, 1970, No 5-6, pp 3-19.

6. Kartsev, M. A. "The M-10 Computer." DOKLADY AKADEMII NAUK SSSR, 1979, 245, No 2, pp 309-312.

2174
CSO: 1863/258

SOFTWARE

UDC 681.3.06:[658.012.011.56:658.7/.8]

OPERATION OF UNIFIED SYSTEM COMPUTERS IN SVS MODE

Moscow MATERIAL'NO-TEKHNICHESKOYE SNABZHENIYE, SERIYA 4: PRIMENENIYE MATEMATICHESKIKH METODOV, VYCHISLITEL'NOY TEKHNIKI I ORGTEKHNIKI V MATERIAL'NO-TEKHNICHESKOM SNABZHENII in Russian No 5, May 80 (manuscript received 17 Jan 80) pp 10-11

[Abstract prepared by G. S. Gradov, deputy department chief, USSR Gossnab Main Computer Center, and V. K. Burmistrov, senior scientific associate of the NIIMS]

[Text] Operation of YeS OS in the Mode of Multiprogramming with a Variable Number of Tasks that Jointly Use Virtual Storage (SVS)

Associates of the USSR Gossnab Main Computer Center and the NIIMS [Scientific Research Institute of Economic and Organization of the Supply of Materials and Equipment] have generated a specific version of YeS OS in the SVS mode for the YeS-1055 computer. The logic of the operation of YeS OS in this mode affords effective support in building complex software systems for large-scale information problems for the USSR Gossnab, such as: interoutput balances, planning of warehouse facilities, bookkeeping reporting and other tasks of the USSR Gossnab ASU.

Operation of YeS Computer in SVS Mode

The YeS OS operating system of the SVS mode is a practical implementation of the concept of virtual storage in YeS computers. It can be used on "Series-2" computers as well as on all computers with similar principles of operation.

The segment-page method of organization of virtual storage is used in YeS OS of the SVS mode. Segment size is 64K bytes and page size is 2K bytes. The beginning address of a segment is an aliquot of its size. The beginning address of a page is an aliquot of its size. The maximal size of virtual storage may reach 16M bytes.

Actual storage holds the pages of virtual storage in use at a given time. Unused pages are stored in external storage in the system data set SYSL PAGE.

The software part that defines the page replacement algorithm is called the replacement algorithm.

Two replacement algorithms are provided in the SVS mode: the standard (LRU--least recently used) and the modified. Sometimes the guaranteed availability of a page of virtual storage may be required in actual storage in the course of a certain time. For this, pages are fixed that are not replaced. Pages are fixed for a short or a long time.

Virtual addresses are converted into real by both hardware and software:

the hardware feature is the dynamic address translation feature (DAT);

the software features are address translation tables: segment tables and page tables.

8545
CSO: 1863/245

EXCERPTS FROM 'ALGORITHMS AND PROGRAMS', FEBRUARY 1981

Moscow ALGORITMY I PROGRAMMY in Russian No 2, Feb 81 pp 1-132

[Excerpts]

475. Simurzin, V. N., "Selection of Rectangular Network by Specified Irregular Networks," in the book "Osnovnyye metody matematicheskoy geologii i resul'taty issledovaniy" [Basic Methods of Mathematical Geology and Results of Research], Yakutsk, 1980, pp 112-118, bibl. of 7 titles.

Gives an example of selecting a rectangular network according to the irregular network of seismic stations of the Sredne-Botuobinskoye gas deposits on the M-222 computer.

584. Bezruk, I. A. and Safonov, A. S., "Principles of Constructing an Automated System to Process Data Obtained by Using Digital Electric Geophysical Exploration Stations," PRIKL. GEOFIZIKA, All-Union Scientific Research Institute of Geophysical Methods of Prospecting, 1980, issue 98, pp 93-102, bibl. of 7 titles.

The basic problem is solved in three stages: preprocessing, the processing proper and interpretation. Basic properties of the EPAK system have been formulated: adaptation of the system on all basic types of computers; simplicity of including new programs in the system, simplicity of operation and introduction of the system.

585. Zabelin, V. A. and Pyatkin, V. P., "Identification of Checkpoints in Aerospace Images," in the book "Matematicheskiye i tekhnicheskiye problemy obrabotki izobrazheniy" [Mathematical and Technical Problems of Image Processing], Novosibirsk, 1980, pp 50-59, bibl. of 4 titles.

Identification of checkpoints is reduced to determining local coincidence of fragments of images using a program in the FORTRAN-TsERN language with a size of about 100 statements. Average operating time for a reference of 20 x 20 elements and a search field of 50 x 50 elements is 3 minutes taking image rotation into account.

586. Lukasheva, O. E.; Pyatkin, V. P. and Shevelev, S. L., "Systems Software for an Image Processing Complex," in the book "Matematicheskiye i tekhnicheskiye problemy obrabotki izobrazheniy", Novosibirsk, 1980, pp 84-94, bibl. of 4 titles.

The KOIZ image processing complex includes three BESM-6 computers operating with a common field of disk storage. The FORTRAN, MADLEN and PASCAL languages were used to implement the image processing complex software.

589. Sidorova, V. S., "Textural Analysis of Aerospace Images by Computer," in the book "Matematicheskiye i tekhnicheskiye problemy obrabotki izobrazheniy", Novosibirsk, 1980, pp 30-36, bibl. of 2 titles.

Describes FORTRAN programs for classification of objects by two systems of textural features based on the image processing complex of the Center for Geoinformation Processing.

590. Toropushina, Ye. F., "Raising the Efficiency of Primary Processing of Tiltmeter Observations," in the book "Seysmichnost' i sovremennyye dvizheniya zemnoy kory vostochnoy chasti Baltiyskogo shchita" [Seismicity and Modern Movements of Crust of Eastern Part of Baltic Shield], Apatity, 1980, pp 108-111, bibl. of 1 title.

Described are the FORTRAN programs BULL-2 and BULL-3 that speed up the process of preparing materials of tiltmeter observations for harmonic analysis.

591. Voronin, Yu. A. and Gradova, T. A., "REGION-kompleks programm dlya postanovki i resheniya zadach rayonirovaniya" [REGION Complex of Programs for Statement and Solution of Problems of Regionalizing], Novosibirsk, 1980, 46 pages, Preprint/ Siberian Branch of the USSR Academy of Sciences Computer Center, No 258, bibl. of 3 titles.

REGION includes two subsystems: HOMREG and DIAREG (homogeneous and diagnostic regionalizing) and the complex of auxiliary programs PREDAT--preliminary data processing in FORTRAN.

706. Pokatayev, A. Yu., "Optimal Organization of Computation of Full Normed Gradients," PRIKL. GEOFIZIKA, All-Union Scientific Research Institute of Geophysical Methods of prospecting, 1980, issue 98, pp 133-136, bibl. of 3 titles.

Author presents original construction of computational process using direct access external storage to store intermediate results. The algorithm has been implemented on the BESM-6 computer.

789. Mikhaylov, N. N. and Yanitskiy, P. A., "Numerical Analysis of Dynamics of Water Saturation in Forming Zone of Penetration in Productive Strata," PRIKL. GEOFIZIKA, All-Union Scientific Research Institute of Geophysical Methods of Prospecting, 1980, issue 98, pp 168-178, bibl. of 13 titles.

Authors describe a physicomathematical model of penetration of filtrate of drilling mud in petroleum and gas bearing strata. Computations were implemented on a YeS-1040 c mputer.

795. Gubin, P. I., "Software for Processing Frame Video Information (in DOS on YeS Computers)," in the book "Trudy MFTI. Seriya Aerofizika i prikladnaya matematika" [Transactions of the Moscow Physicotechnical Institute: Series on Aerophysics and Applied Mathematics], Moscow, 1980, pp 125-127.

The SMOOKI program package for processing aerospace video information supports putting information into a computer from ML [magnetic tape], storage of it, statistical processing and reflection of processing results in various forms.

812. Kondrat'yev, O. K. and Orlov, V. P., "Diffraction Transformation of Seismograms in the Method of Transmitted Composite Waves," PRIKL. GEOFIZIKA, All-Union Scientific Research Institute of Geophysical Methods of Prospecting, 1980, issue 98, pp 42-53, bibl. of 13 titles.

Algorithm for construction of dynamic deep sections by seismograms of method of transmitted composite waves (MPOV) has been implemented on the BESM-4 computer. Application of D-transformation in the MPOV eliminates correlation and determination of travel time curves of waves from the procedure for construction of a deep section. To obtain one path of a section, two-three minutes are required.

813. "Method of Calculation of Composition and Properties of Transfer of Heterogeneous Systems," Ye. V. Samuylov, I. B. Rozhdestvenskiy, N. N. Tsitelauri and N. A. Budko, in the book "Osnovnyye metody matematicheskoy geologii i resul'taty issledovaniy", Yakutsk, 1980, pp 78-82, bibl. of 13 titles.

814. "Computer Modeling of Process of Processing of Frequency Modulated Signals in Devices on Surface Acoustic Waves," Ye. Ya. Bershadskiy, V. A. Vinogradova, V. K. Sirotko and A. P. Fedorov, ELEKTRON. TEKHNIKA. SER. 10. MIKROELEKTRON. USTROYSTVA, 1980, issue 5(23), pp 61-66, bibl. of 5 titles.

Technique of modeling of formation and compression of frequency modulated signals in devices on surface acoustic waves has been implemented on the M-222 computer. Calculation time when 1024 points are specified is about two minutes.

8545
CSO: 1863/246

ADDITIONAL EXCERPTS FROM 'ALGORITHMS AND PROGRAMS', FEBRUARY 1981

Moscow ALGORITMY I PROGRAMMY in Russian No 2, Feb 81 pp 1-132

[Excerpts]

398. Gavrilov, G. K., "Efficient Programming Technology," VOPR. RADIOELEKTRON. SER. EVT, 1980, issue 8, pp 135-141, bibl. of 8 titles.

Recommendations on imrpoving labor productivity of programmers and improving the quality of programs.

485. Brodskaya, I. M. and Kamynin, S. S., "Algorithm for Recognition of Partially Visible Objects," Moscow, 1980, 34 pages, Preprint/Academy of Sciences of USSR, IPM [Institute of Applied Mathematics], No 107, bibl. of 8 titles.

Size of training program is 1110 single-address instructions, and that of the recognition program is 3124. About 14500 words of main storage is required to operate the program.

505. Pikel'ner, B. L. and Romanov, V. D., "Dynamic Display of Multiprogramming in YeS OS," EKELTRON, TEKHNIKA. SER. 9. EKONOMIKA I SISTEMY UPR., 1980, issue 3(36), pp 31-32, bibl. of 4 titles.

A method of obtaining a graphic presentation of the multiprogramming factor as a function of time has been implemented in the PL/1 language. Program size is 76K bytes, and time for construction and printout of daily diagram is three to five minutes.

511. Zaytseva, Zh. N., Yablonskaya, O. V. and Bodryagin, V. I., "Procedure for Selection of Printed Conductors for Layout," VOPR. RADIOELEKTRON. SER. EVT, 1980, issue 8, pp 124-126, bibl. of 3 titles.

PL/1 program for selection of arrangement of conductors for layout, which supports minimization of conflict situations in designing printed circuit boards, with a size of 232 statements takes 22 seconds of machine time.

513. RADIOTEKHNIKA: RESP. MEZHVED. NAUCH.-TEKHN. SB., Khar'kov Institute of Radioelectronics, issue 55, Khar'kov, Vishcha shkola, 1980, 109 pages, bibl. at end of articles.

From contents: V. M. Bezruk, V. A. Omel'chenko and N. A. Federov, "Comparative Analysis of Adaptive and Correlation Methods of Receiving Discrete Signals," pp 3-11; V. A. Omel'chenko et al., "Recognition of Random Signals Based on Digital Analyzers of the Spectrum," pp 11-17; Yu. A. Kichigin, Z. N. Muzyka and S. A. Sokolov, "Study of Statistical Characteristics of Signal-to-Noise Ratio at Output of Radio Receiver with Nonlinear Interaction of Signal and Noise in Its First Cascades," pp 18-24; S. F. Simovskaya and O. M. Lebedeva, "Analysis of Correlation Function of Random Radio Signal by Method of Digital Modeling on Carrier Frequency," pp 79-84.

517. Brodskiy, A. Ya. and Yevseyeva, L. M., "Software for Data Retrieval System for Scientific Research Institute," ELEKTRON. TEKHNIKA. SER. 9. EKONOMIKA I SISTEMY UPR., 1980, issue 3(36), pp 26-29.

528. Alekseyev, K. V., "Subsystem for Automated Synthesis of Checking Experiments for Discrete Devices," TEORIYA KONECH. AVTOMATOV I YEYE PRIL., Academy of Sciences of Latvian SSR, Institute of Electronics and Computer Technology, 1980, issue 11, pp 79-91, bibl. of 4 titles.

Size is about 3000 instructions. Test for checking of circuit of 23 elements with 7 inputs, 7 signal branching points and 1 output, consisting of 14 sets, obtained in 6 minutes and 40 seconds.

Key words: methodology, FORTRAN-4, Nova 2/10, checking experiments, discrete devices.

529. Grishina, I. B., "'Library Statistics' Program Package," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 7(319), pp 74-75, bibl. of 3 titles.

Describes the FORTRAN program package "Library Statistics" for acquisition and storage of statistics on use of programs from common libraries.

562. Kol'dyayev, V. I., "Kinetics of Charge Injection in MDP [metal-insulator-semiconductor] Transistors with Floating Gate," ELEKTRON. TEKHNIKA. SER. 3. MIKROELEKTRON., 1980, issue 2(86), pp 14-20, bibl. of 17 titles.

563. Kol'dyayev, V. I., "Kinetics of Charge Relaxation from Insulated Layers of Polycrystalline Silicon," ELEKTRON. TEKHNIKA. SER. 3. MIKROELEKTRON., 1980, issue 3(87), pp 14-19, bibl. of 11 titles.

Difficulties of depolarization of insulated conducting layers, prerequisites of kinetics of charge relaxation. Mathematical model of phenomenum has been constructed on the assumption of Schottky and Fowler-Nordheim emission. Model of charge spread has form of a system of algebraic differential equations that are solved with a FORTRAN program. Computation time of one integration for t from 0 to 100 years is about 2 minutes.

564. Kondrat'yev, I. I., "Mathematical Processing of Gamma Spectra of Samples of Aerosols Using Correlation Filtering Method," TR./DVNII, 1980, issue 91, "Problems of Hydrometeorology of Eastern Siberia, the Far East and the Pacific Ocean," pp 78-83, bibl. of 5 titles.

567. Malinovskaya, Ye. V. and Malyshev, Yu. A., "Calculation of Resistance of Layer of Pinch-Resistor of Semiconductor Microcircuits," ELEKTRON. TEKHNIKA. SER. 3. MIKROELEKTRON., 1980, issue 3(87), pp 20-27, bibl. of 6 titles.

Algorithm forms basis of FORTRAN program that performs either "single" analyses, i.e. calculation of resistance of layer with fixed parameters of technological process (time is 2-3 seconds) and voltages, or calculation of dependency of resistance of layer on minimal back bias and voltage on resistor with fixed technological parameters (time is 1-2 minutes). Storage size is 22K bytes.

568. Man'kin, I. A., Usherovich, B. L. and Kontorin, Yu. F., "Program for Calculation of Nonlinear Conditions of LBV [Traveling Wave Tuve] on Chain of Connected Resonators," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 7(319), pp 75-76, bibl. of 6 titles.

Described is FORTRAN program for calculation of nonlinear output characteristics of LBVR (efficiency factor, gain factor, wave phase) in a band of frequencies as a function of geometric and electric parameters of instrument. Calculation time for three points of operating range of systems with number of resonators not exceeding 20 does not exceed 15 minutes.

571. Mirgorodskiy, Yu. N.; Myachin, G. M. and Rudenko, A. A., "Program for Calculation of Dynamic Characteristics of Silicon Planar Schottky Diode," ELEKTRON. TEKHNIKA. SER. MIKROELEKTRON., 1980, issue 3(87), pp 90-92, bibl. of 5 titles.

Length of FORTRAN program is 550 conditional statements, reserved size of storage is 32 sheets of main storage. Calculation time of one point on pulse characteristic of planar Schottky diode is 20 seconds.

572. Mirgorodskiy, Yu. N.; Myachin, G. M. and Rudenko, A. A., "Program for Calculation of Static VAKh [Volt-Ampere Characteristic] of Silicon Planar Schottky Diode," ELEKTRON. TEKHNIKA. SER. 3. MIKROELEKTRON., 1980, issue 2(86), pp 98-100, bibl. of 2 titles.

575. Pasynkov, I. G. and Skoryupina, I. V., "Interactive Subsystem for Correcting Double-Sided Printed Circuit Boards Designed by Using 'Avtomat-2' System," Moscow, 1980, 13 pages, Preprint/IAE, No 3312/16, bibl. of 6 titles.

"Correction" subsystem for editing printed circuit boards in interactive mode using VT-340 type alphameric display. Program package is written in FORTRAN-GDR and MADLEN languages.

576. "Application of Method of Finite Elements to Calculation of Temperature Fields in Assemblies of Electronic Instruments," by I. M. Bleyvas, A. I. Zhbanov, V. S. Koshelev and V. N. Shevtsov, ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 7(319), pp 72-74, bibl. of 1 title.

Average time for solving one problem is one to five minutes.

598. Dydychkin, A. Ye., "Method of Calculation of References of Typewritten Characters with Computer," VOPR. RADIOELEKTRON. SER. EVT, 1980, issue 8, pp 167-174, bibl. of 2 titles.

Describes a FORTRAN program for automatic construction of references. Program run time is proportional to number of points approximating the character and ranges from 3 to 12 minutes.

599. Dydychkin, A. Ye. and Kalinin, A. V., "Modeling the Process of Generating Typewritten Print," VOPR. RADIOELEKTRON. SER. EVT, 1980, issue 8, pp 161-166, bibl. of 4 titles.

Method of reducing sample size for teaching and assessing validity of automatic reading machines based on using FORTRAN program to generate binary images of typewritten print.

615. Yasenovets, A. V., "Software for Formatting and Editing of Text for Wang Minicomputer," in the book "Tipovoye proyektirovaniye vtoroy ocheredi otraslevykh podsistem ASPR" [Standard Design of Second Phase of Sector Subsystems for Automated System of Plan Calculations], Moscow, 1980, pp 108-111.

643. Litavrina, V. N. and Nikitin, S. A., "Language for Describing Topology of SVCh GIS [Super-High-Frequency Hybrid Integrated Circuits]," ELEKTRON. TEKHNIKA. SER. 10. MIKROELEKTRON. USTROYSTVA, 1980, issue 5(23), pp 53-56, bibl. of 1 title.

Language for coding SHF microcircuits is used in a system for automated design of topologies of integrated circuits and forms a part of the GRAF language for description of graphic information.

682. Kazanskiy, A. V.; Khaykin, A. P. and Trusenkov, S. T., "Application Program Package for Man-Machine Analysis and Forecast of Temperature of Surface Layer of Ocean," TR./DVNII, 1980, issue 91, "Problems of Hydrometeorology of Eastern Siberia, the Far East and the Pacific Ocean," pp 39-43, bibl. of 9 titles.

683. Yukhnovets, B. P. and Burzayeva, Ye. I., "Checking Topology and Complex Patterns in Designing Hybrid Film Microcircuits," TR./NIITEPLOPRIBOR, 1980, collection 89, "Technology of Manufacture and Automated Design of Transducers of Heat and Power Parameters," pp 56-63, bibl. of 4 titles.

Method of checking topological patterns for minimally permitted dimensions has been implemented in a system for automated design of film hybrid microcircuits--SAPR "PLATA"--MSSP. Program run time when checking topology containing 400 points on the M-4030 computer is 15 seconds.

692. Tikhonov, Yu. V., "Software for 'INTRAS' Interactive Graphic System," in the book "Trudy MFTI. Seriya Aerofizika i prikladnaya matematika" [Transactions of the Moscow Physico-Technical Institute: Series Aerophysics and Applied Mathematics], Moscow, 1980, pp 144-146, bibl. of 3 titles.

697. Afanasov, S. G.; Kochetkov, V. I. and Cherednik, V. I., "Second Zone of Oscillator with Brake Field," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 7(319), pp 21-25, bibl. of 6 titles.

Results of theoretical research of nonlinear interaction of electron flow with high-frequency field of working interval of oscillator with brake field (GTP) in second zone of oscillation. System of equations for nonlinear unidimensional theory of GTP has been solved on the BESM-6 computer by the Cauchy-Euler method.

701. Karimberdiyeva, S., "Some Studies of Daily Course of Components of Heat Balance of Active Surface," TR./SREDNEAZ. REGION. NII, 1980, issue 68(149), "Problems of Hydrodynamics of Atmosphere," pp 32-42, bibl. of 14 titles.

Describes algorithm for solving equation of heat balance on BESM-6.

702. Mazilov, V. N. and Kashik, S. A., "Application of Clastic Analysis in Mineralogical Studies," in the book "Osnovnyye metody matematicheskoy geologii i resul'taty issledovaniy" [Basic Methods of Mathematical Geology and Results of Research], Yakutsk, 1980, pp 66-70, bibl. of 2 titles.

703. Mayorov, S. A. and Rudenko, A. A., "Method of Calculation of Dynamic Characteristics of MDP [Maximum Dynamic Error] of Capacitor in Two-Dimensional Approximation on Large Signal," ELEKTRON. TEKHNIKA. SER. 3. MIKROELEKTRON., 1980, issue 2(86), pp 79-83, bibl. of 4 titles.

Example of calculation on BESM-6 computer of nonstationary distributions of potential and carriers of charge for solving problem of calculation of dynamic VAKh [volt-ampere characteristics] and VFKh [expansion unknown] of MDP of capacitor in two-dimensional approximation.

704. Mirgorodskiy, Yu. N. and Rudenko, A. A., "Method of Calculating Static Characteristics of Silicon Planar Schottky Diode," MIKROELEKTRON. I POLUPROVODNIKOVYYE PRIBORY: SB. STATEY, 1980, issue 5, pp 194-199, bibl. of 10 titles.

Calculation of volt-ampere characteristics (VAKh) of Schottky planar diode is based on solution of boundary problem of nonlinear system of equations of continuity and Poisson in two-dimensional area for the case when width of the area of space charge at Schottky contact is small compared to characteristic linear dimension of diode. Cross-section of Schottky diode is covered by spatial network with number of assemblies being about 1000. VAKh calculation time for Schottky diode on BESM-6 computer is 8 minutes.

708. Vekshina, Ye. V.; Skorik, V. A. and Fursin, G. I., "Machine Modeling of Shift Register with Thyristors with Plasma Bond," ELEKTRON. TEKHNIKA. SER. 3. MIKROELEKTRON., 1980, issue 2(86), pp 63-66, bibl. of 5 titles.

Shift register with instruments with plasma bond (PPS) is base element for development of logic circuits, parallel adders, triggers and storage. Model of PPS for computer design of transistorized LSI circuits allows determining extremal parameters of a shift register.

710. Nechayev, A. M.; Rubakha, Ye. A. and Sinkevich, V. F., "Modeling Failures and Machine Tests in Studying the Reliability of Power SVCh [Microwave] Transistors," MIKROELEKTRON. I POLUPROVODNIKOVYYE PRIBORY: SB. STATEY, 1980, issue 5, pp 98-117, bibl. of 16 titles.

Discussed are machine models of main types of failures of power microwave transistors: simulation model of thermal breakdown and physico-statistical model of electrodiffusion failures. Up to 100 hours of BESM-6 computer operation are needed to implement the machine tests of transistor structures.

772. Pogodin, A. Ye., "Ways of Raising Computer Throughput," VOPR. RADIOELEKTRON. SER. EVT, 1980, issue 8, pp 131-134, bibl. of 3 titles.

Raising throughput of software systems built with YeS computer facilities through organization and arrangement of data on MD [magnetic disk] storage units to reduce data access time.

792. Sunduchkov, K. S. and Leshchenko, Yu. I., "Automation of Synthesis of Structural Circuits of Low-Noise Receiver-Amplifier SVCh [Microwave] Devices," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 7(319), pp 64-69, bibl. of 14 titles.

796. Kirillov, D. A.; Naumenko, V. I. and Skorokhodov, O. V., "Development of Hierarchical Models in the 'Poisk-1' Information System," VOPR. RADIOELEKTRON. SER. EVT, 1980, issue 8, pp 3-9, bibl. of 4 titles.

The Poisk-1 documentary data base management system supports efficient retrieval in large data bases. Language for defining hierarchical structures is suggested and language for manipulating systems is expanded.

810. Belyavskiy, Ye. D. and Gel'ner, V. V., "Numerical Analysis of Nonlinear Distortion of Shape of Nonperiodic Signal in LBV [Traveling Wave Tube]," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 7(319), pp 69-72, bibl. of 5 titles.

Technique for strict (unidimensional) calculation with BESM-4 computer of shape of nonperiodic signal when it passes through LBV, on basis of which is made numeric analysis of nonlinear distortion of short pulses in LBV in nonlinear mode with regard to nonstationary effects.

834. "Modular Construction of Software for Automated Process Control System for Production of Integrated Circuits," by V. M. Gudkov, V. I. Nadtochiy, A. L. Shabalin and T. P. Shvankova, ELEKTRON. TEKHNIKA. SER. 9. EKONOMIKA I SISTEMY UPR., 1980, issue 3(36), pp 29-31, bibl. of 3 titles.

Modular software construction achieves invariance of it with respect to problems and objects of control. Implementation of principles in system for monitoring quality of process gases, developed on the base on the Elektronika-100I minicomputer, has shown their high efficiency.

835. Molchanova, L. P. and Ostrovskiy, V. I., "Algorithm for Identification of Process Defects in Microcircuits," ELEKTRON. TEKHNIKA. SER. 10. MIKROELEKTRON. USTROYSTVA, 1980, issue 5(23), pp 49-52, bibl. of 2 titles.

Algorithm has been implemented in the form of two programs for the Elektronika-100I computer. Time for analysis of one microcircuit with dimensions of 1 x 1 cm is about 7.5 seconds.

836. Osetinskiy, L. G. and Osetinskiy, M. G., "Software Controlled Process System for Manufacture of Photographic Masks for Large-Scale Integrated Circuits," ELEKTRON. TEKHNIKA. SER. 9. EKONOMIKA I SISTEMY UPR., 1980, issue 3(36), pp 23-25, bibl. of 4 titles.

Organization of storage on ML [magnetic tape] of control information for generators of images--file OS [operating system] that supports control of photocomposition equipment from the Elektronika-100I minicomputer.

860. Astsaturov, R. M.; Nikolayeva, G. V. and Sigalov, G. G., "Study of Algorithms for Control of Heirarchical Storage of Microprograms," VOPR. RADIOELEKTRON. SER. EVT, 1979, issue 12, pp 3-14, bibl. of 8 titles.

Using analytic and simulation models, study is made of a number of algorithms that support fetching microinstructions for execution from fast and slow section of control storage.

879. Samarskiy, A. S.; Pykhtin, V. Ya. and Vasilevich, V. V., "Estimating Complex Throughput of Computers by Analytic Methods," VOPR. RADIOELEKTRON. SER. EVT, 1979, issue 12, pp 69-79, bibl. of 8 titles.

Analytic model for estimating complex throughput, basic elements of which are the processor and selector channels; procedure for calculating level of multiprogramming with regard to computer resources. Gives an example of calculating throughput for the YeS-1022, -1033 and -1035 medium class computers.

880. Sigalov, G. G. and Kur'yanova, N. I., "Estimating Computer Operation with Separation of Input-Output from Job Processing," VOPR. RADIOELEKTRON. SER. EVT, 1979, issue 12, pp 41-47, bibl. of 4 titles.

Technique of using model of computer operation under conditions of separating input-output from process of job processing together with estimate of maximum possible factor of combining operation of devices for selection of number of input-output devices, degree of multiprogramming and maximum size of job queue on disk.

882. "Structure of Collective-Use Automated Design System," by A. V. Grekovich, A. A. Bozhko, A. B. Trofimov and N. A. Federov, VOPR. RADIOELEKTRON. SER. EVT, 1979, issue 12, pp 53-59, bibl. of 2 titles.

Principles for development of structure of INTER-SAPR system, problems of organization of exchange of application program facilities, trilevel principle of system coordination.

887. Shklyar, V. B. and Yatsevich, P. F., "Model of Functioning of Heirarchical Storage of Microprograms," VOPR. RADIOELEKTRON. SER. EVT, 1979, issue 12, pp 15-19, bibl. of 2 titles.

Probabilistic model has been built on basis of Markov model of random process.

8545
CSO: 1863/246

EXCERPTS FROM 'ALGORITHMS AND PROGRAMS', MARCH 1981

Moscow ALGORITMY I PROGRAMMY in Russian No 3, Mar 81 pp 1-128

[Excerpts]

920. "Territorially Distributed Multimachine Collective-Use Computer Center," by G. I. Marchuk, Yu. I. Yeremin, G. I. Karpachev et al., Novosibirsk, 1980, 59 pages, (Preprint/Siberian Branch of USSR Academy of Sciences Computer Center, 245).

1036. Zarin'sh, A. Ya., "Noise Resistant Method of Filtration of Satellite Laser Observations," NAUCH. INFORM./AN SSSR. ASTRONOM. SOVET, AN LATVSSR. RADIOASTROFIZ. OBSERVATORIYA, 1980, issue 44, "Use of Satellite Observations in Geodesy and Geophysics," pp 40-43, bibl. of 2 titles.

1076. Vitrichenko, E. A.; Prokhorov, A. M. and Trushin, Ye. V., "Metody izgotovleniya astronomicheskoy optiki" [Methods of Manufacturing Astronomical Optics], Moscow, Nauka, 1980, 196 pages, above the title: Academy of Sciences of USSR, Institute of Space Research, bibl.: pp 138-142.

Describes the American CAOS and CCP and the domestic ZEBRA automated process systems in the FORTRAN language. The ZEBRA system consists of two subsystems: checking and technology. The checking subystem has been implemented by the Hartman method. Gives texts of programs of instructions for operator and checking examples.

1084. Mironov, N. T. and Bogatyrev, K. A., "Calculation of Ephemeris of Satellites for Observations with First-Generation Laser Ranger," ASTROMETRIYA I ASTROFIZIKA: RESP. MEZHVED. SB./AN UkSSR, GL. ASTRON. OBSERVATORIYA, 1980, issue 42, pp 83-89, bibl. of 4 titles.

1097. Trifonov, M. M., "Translator that Forms Programs of Base Elements of GIS SVCh [Microwave Hybrid Integrated Circuits] Based on Experimental Data," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 8(320), pp 67-69, bibl. of 7 titles.

Describes FORTRAN program package that by matrices of scattering S or matrices of conductivity Y, specified in band of frequencies, forms program of base element in autocode of BESM-6 MADLEN.

1098. Fialovskiy, A. T.; Zaytsev, S. V. and Mikheyev, A. G., "Calculation of Constants of Propagation and Diffraction Losses of Higher Type of Waves in Nonsymmetrical Band Lines," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 8 (320), p 66, bibl. of 3 titles.

Iteration algorithm with number of variations of field by thickness of dielectric basis greater than or equal to 1 in the FORTRAN language. Average calculation time of one variant is about 0.08 second.

1115. "Software for Subsystems to Process Images and Charts in Isolines and Numeric Data," Moscow, 1980, 68 pages, (ALGORITMY I PROGRAMMY/AN SSSR. BIEMS, GIVTS ASU - GEOLOGIYA, issue 6 (41)).

Described are FORTRAN programs for systems that process multispectral photographs, halftone images, binary images of schemes of lineaments and erosion networks, and charts of relief in isolines and numeric data.

1146. Batyrev, Ye. V.; Ocheretyannyy, A. N. and Sazonov, A. A., "On-Line Monitoring of Operating Capability and Diagnostic Checking of Systems for Synchronization of Checking and Measuring Systems," ELEKTRON. TEKHNIKA. SER. 3. MIKROELEKTRON., 1980, issue 4 (88), pp 65-74, bibl. of 3 titles.

Algorithm for checking efficiency of M-6000 computer Assembler language takes no more than 300 instructions, and execution time including printout is no more than 2 seconds. The algorithm for diagnostic checking has been implemented in the BASIC language, and execution time with printout of results is no more than 5 minutes.

1214. Rebrik, S. B., "Movement of Eyes and Processes of Organization of Sensorimotor Activity," ERGONOMIKA. TR./VNIITEKHN. ESTETIKI, 1980, issue 19, "Studies of Functional Structure of Actuating Activity," pp 75-89.

Studies to define the functional relationship between processes of organization of instrumental activity and eye movement activeness accompanying it were performed on the M-6000 computer.

1215. Sirotkina, Ye. B. and Apipova, A. I., "Study of Temporal, Preciseness and Spatial Characteristics of Movement under the Effect of Stresses," ERGONOMIKA. TR./VNII TEKHN. ESTETIKI, 1980, issue 19, "Studies of Functional Structure of Actuating Activity," pp 38-43.

Study of method of microstructure analysis with M-6000 computer to find out selective effect of medicinal preparation of (kavinton) on structural components of whole activity.

1224. Grigor'yev, A. D.; Sipayev, S. A. and Yankevich, V. B., "Computer Design of Ring-Type Resonators," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 8 (320), pp 66-67, bibl. of 2 titles.

Described is an algorithm for calculation of working E-type of oscillations in ring-type resonators (coaxial) using condition of orthogonality of eigen functions. Results of solving test-problems on the BESM-6 are given.

1237. "Operating Instructions for the CRJE System for Local Displays of a YeS-7906 Complex," by R. S. Triandofilidi, Moscow, 1980, 89 pages (METOD. REKOMENDATSII PO PROGRAMMIR. NA YES EVM/AN SSSR. TSEMI, issue 18), bibl. of 4 titles.

Instruction on preparing jobs for the system, examples of use of job input commands, job editing during debugging, obtaining results of an executed job and conversation with central computer operator.

1242. Przhiyalkovskiy, V. V. and Lomov, Yu. S., "Tekhnicheskiye i programmnyye sredstva Yedinoy sistemy EVM (YeS EVM-2)" [Unified Computer System Hardware and Software (YeS Computer-2)], Moscow, Statistika, 1980, 230 pages, bibl. of 21 titles.

1249. Andreyev, G. G. and Koptseva, N. N., "Machine Method of Smoothing Out Average Photo Tone over Area of Aerial Photographic Image," TR./GOS. N.-I. TSENTR IZUCH. PRIROD. RESURSOV, 1980, issue 10, "Some Results of Natural Resource Research Using Aircraft and Polygon Facilities," pp 38-41, bibl. of 1 title.

Flowchart of program for YeS-1022 computer to correct photo tone of aerial photographic images to eliminate distortions stemming from nonuniformity of function of distribution of illumination of lens.

1263. Arifov, P. U.; Zakirov, T. A. and Nazirov, A., "Construction of Multimachine System to Automate Experiments on Emission Electronics," VOPR. KIBERNETIKI, UzSSR Academy of Sciences, Institute of Cybernetics and Computer Center, 1980, issue 110, VT V UPR.: SB. NAUCH. TR., pp 8-13, bibl. of 4 titles.

Problems of developing a multimachine system based on the YeS-1022, the Elektronika K-200 and CAMAC standard equipment using the principles of unified bus system for exchange of information.

1273. Platonov, A. V., "Complex of Problems of Accounting of Traffic of Physical Assets," VOPR. RADIOELEKTRON. SER. ASU, 1980, issue 1, pp 57-63, bibl. of 3 titles.

1311. Vayda, F. and Chakan', A., "Microcomputers," translated from Hungarian, Moscow, Energiya, 1980, 359 pages, bibl. of 32 titles.

Organization, principles of operation and fields of application of microprocessors Intel 8008, 8080, 3000, PPS25, PPS4, MCS4, and questions of designing microcomputers based on microprocessors.

8545
CSO: 1863/246

EXCERPTS FROM 'ALGORITHMS AND PROGRAMS', APRIL 1981

Moscow ALGORITMY I PROGRAMMY in Russian No 4, Apr 81 pp 1-112

[Excerpts]

1462. Batalov, A. F.; Bobkov, V. A. and Ovchinnikova, L. S., "Program Package for Operating with the UGD-43-1 Graphic Display," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, p 112, bibl. of 4 titles.

The BOB program package is designed for operating with the UGD-43-1 display, working under control of the Elektronika-100I minicomputer connected to a BESM-6 by communication channel in DISPAK DOS.

1470. Bogdanov, N. B. and Kupenova, T. N., "Programs for Analytic Continuation of Holomorphic Functions beyond Linear," Dubna, 1980, 13 pages (SOOBSHCH./OIYAI, No R11-80-621), bibl. of 4 titles.

NAC program package is written in FORTRAN-4 for the CDC-6500 (with conventional precision) and text is given for the YeS and IBM computers (with double precision).

1486. Ashkenazi, I. D. and Korolev, A. N., "Program for Calculating Dispersion Characteristics, Wave Impedance and Width of Microband Line," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 1, pp 122-126, bibl. of 5 titles.

Described is the FORTRAN program STRIP1 for calculation of base elements in a system for machine design of microwave hybrid integrated circuits. Calculation time for one set of input parameters varies from a hundred microseconds to three milliseconds with a known linewidth; with a known wave impedance, it is three to fivefold greater.

1488. Bunimovich, B. F. and Sharayevskiy, Yu. P., "Program for Analysis of Frequency Responses of Multisectional LBVO [expansion unknown]," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 1, pp 119-121, bibl. of 3 titles.

Description of FORTRAN program to calculate time lag and coupling impedance of sections of a slow-wave system and gain factor for LBVO with various values of geometric and electrical parameters, varying with the design. Calculation time for one frequency response depends on number of sections and number of discrete points within the frequency band (one frequency point is about one second).

1492. Kabakov, L. T. and Rukomoyeva, A. I., "Calculation of Coupling of Two Rectangular Waveguides through a Rectangular Aperture in a Common Wide Wall of Finite Thickness," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 1, pp 121-122, bibl. of 3 titles.

The program has been implemented in the FORTRAN language. Calculation time for one alternative is no more than 20 seconds.

1494. "Program Package, 'Analysis of Linear Transistor Microwave Amplifiers'," by V. Yu. Yakovenko, S. V. Dvornikov, V. G. Volnistyy and N. N. Perestyuk, ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 2, pp 112-114, bibl. of 4 titles.

The program package was written by using the FORTRAN-4 and Assembler languages. Main storage size is 64K bytes. Analysis time for one alternative of a two-stage transistor amplifier at 10 frequency points is 2 to 3 minutes.

1496. Nemirovskiy, E. E., "Reed-Solomon Codes, Resistant to Phase Inversion in Channel," ELEKTRON. TEKHNIKA. SER. 10. MIKROELEKTRON. USTROYSTVA, 1980, issue 3 (21), pp 60-63, bibl. of 3 titles.

Described is a FORTRAN program that includes arithmetic of finite fields and takes about 1.5 minutes of machine time to check for the field GF (2^8).

1498. "Program for Approximate Analysis and Optimization of LBVO [expansion unknown] in Specified Band of Frequencies," by V. G. Borodenko, V. P. Kiryushin, A. S. Krasil'nikov et al., ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 2, pp 105-110, bibl. of 9 titles.

Program was compiled in FORTRAN. Calculation time at one point of band for electric length v = 10 is about 1 second. Problem solving time is 35 minutes.

1525. "Program Package for Sorting Special Type Information," by V. V. Vishin, S. A. Zaytsev, A. M. Maslennikov et al., ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, pp 110-111, bibl. of 2 titles.

1526. Maslennikov, A. M., "Programs for Checking Information Being Processed," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, p 110, bibl. of 2 titles.

1611. "Matematicheskoye obespecheniye gibidnoy vychislitel'noy sistemy 'Rusalka'. Gibridnyy assembler. Gibridnyy dialogovyy otladchik" ["Rusalka" Hybrid Computer System Software: Hybrid Assembler, Hybrid Interactive Debugger], Institute of Control Problems, Moscow, 1980, 63 pages.

The hybrid assembler includes the M-400 computer ASS400A assembler and the hybrid superstructure that supports communication between the converter and interface devices and digital and analog parts of the GVS [hybrid computer system], and operates in the real-time mode.

1613. "Elektronnaya vychislitel'naya mashina M-4030: Posobiye pol'zovatelyu" [M-4030 Computer: User Handbook], by K. N. Belousov, N. V. Plakhotnyy, V. S. Mokhonchuk et al., Kiev, Tekhnika, 1980, 247 pages.

Information needed for independent solution of practical problems on the M-4030 computer. Discussed are questions of M-4030 hardware, job control, library utilities, main operator commands and system messages. Problem solving process is described in Assembler, FORTRAN and PL/1.

1615. "Computer Terminal Software," by A. N. Zakharova, L. S. Ovchinnikova, L. F. Pozdnyakova and T. N. Romanova, ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, p 113, bibl. of 4 titles.

Purpose of the user station (AVP) is to organize input of user problems and sending them over communication line to BESM-6 for solving. User station includes the Elektronika-100I and various peripherals.

1617. "'TEST-3' Program for Checking Serviceability of BESM-6 Equipment for Communication with Minicomputers by Seventh Direction through Computer Switch," by A. N. Zakharova, S. A. Zaytsev, Ye. I. Karmazina et al., ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 2, pp 110-111, bibl. of 7 titles.

See abstract No 1662.

1618. Sakharchuk, V. I. and Serebrennikov, P. I., "Program for Checking Serviceability and Finding Defects in BESM-6 and 'Elektronika 100/16-I' Interface Harware: Engineering Test of Communications," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, pp 105-106, bibl. of 1 title.

1619. Sorokin, A. G., "Subroutine for Drawing Text Characters," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, pp 106-107, bibl. of 5 titles.

Subroutine run time for BESM-6 for one access is independent of length of specified character string and is greater than or equal to 5 seconds.

1620. Bryunin, V. N.; Bulatov, M. Kh. and Galitskiy, A. V., "Automation of Reliability Design of Microelectronic Apparatus with Redundancy," ELEKTRON. TEKHNIKA. SER. 10. MIKROELEKTRON. USTROYSTVA, 1980, issue 3 (21); pp 90-102, bibl. of 9 titles.

Automated system for assessing and optimizing reliability (ASON) of modeling and computation of theoretical functions of reliability of complex systems with various types of redundancy has been implemented on the BESM-6. Described is YaONS specialized language for describing and automating synthesis of mathematical reliability models of redundant structures.

1621. Dashkovskiy, A. A., "Panel Method Calculation of Flow about Wing Profile by Flow of Incompressible Liquid," TR./TsAGI, 1980, issue 2089, "Methods of Calculating Aerodynamic Characteristics of Profiles, Bodies of Revolution and Systems of Bodies: A Collection of Articles," pp 3-18, bibl. of 3 titles.

BESM-6 computer calculation time for a profile whose surface is subdivided into 100 panels is about 2 minutes; for 60 panels, time is 50 seconds; and for 30 panels, time is 16 seconds.

1623. Il'inskiy, A. S. and Zarubanov, V. V., "Program for Calculating Distributions of Currents of Normal Waves of Nonsymmetrical Band Line (MSTRIP)," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 1, pp 126-128, bibl. of 5 titles.

Program for calculating distributions by band of longitudinal and cross currents of basic and higher types of nonradiating waves of partially screened nonsymmetrical band line. Obtaining all graphs took about 2.5 minutes of BESM-6 central processor operation.

1624. Oleynikov, V. I., "Technique for Calculating in Three-Dimensional Approximation Nonlaminar Elliptic Electron Beams in Transit Channels of O and M Types of Microwave Instruments," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 1, pp 51-61, bibl. of 8 titles.

1628. "Machine Design of Electronic Optical Systems Using Optimization Method," by I. I. Golenitskiy, I. B. Grishina, T. P. Kutsevskaya et al., ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, pp 98-105, bibl. of 6 titles.

Examples of solving problems based on a software package that includes a program for analysis of two-dimensional electronic optical systems (EOS), a standard program for optimization for the BESM-6 and developed program modules that interpret user language in which problem is formulated.

1641. Takhtamyshev, G. G., "'Tvist' Modeling Program," Dubna, 1980, 8 pages, (SOOBSHCH./OIYaI, No 1-80-640), bibl. of 12 titles.

Description of program for determining characteristics of magnetic spectrometers and checking programs of geometric reconstruction on CDC-6500 and YeS-1040 computers. The program traces tracks, belonging to the physical process, through the spectrometer and forms a unified array of all tripped wires. Program size (together with user programs and package for histograms) is 55200_8 words.

1643. Shemyakin, V. V. and Krasil'nikov, D. P., "Experimental System for Acoustic Emission Monitoring Based on YeS-1010 Computer: MO [Software]," Moscow, 1980, 24 pages, (Preprint/IAE, No 3333/15), bibl. of 4 titles.

1653. Rabinovich, Ye. B.; Sinitsyna, G. B. and Feynberg, V. Z., "Determining Optimal Parameters of Complex Mark of Matching of Drawings of Integrated Circuits," ELEKTRON. TEKHNIKA. SER. 7. TEKHNOLOGIYA, ORG. PR-VA I OBORUD., 1980, issue 1 (98), pp 73-76, bibl. of 6 titles.

M-222 computer was used to obtain table of optimal parameters of complex marks that are used in settings for automatic matching of IC drawings.

1662. Zakharova, A. N. and Ovchinnikova, L. S., "Program 'Test of Complex of On-Line Graphic Station'," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 1, pp 118-119, bibl. of 1 title.

On-line graphic station (POGr) has been developed on base of Elektronika computer with display and connected to BESM-6 and Videoton VT-340. Program is intended for checking serviceability of POGr hardware and communication line between the Elektronika-100/I and BESM-6 computers. Program is written in SLENG-3 autocode.

1663. Pozdnyakova, L. F., "Program for Checking Serviceability of Card Input Unit (UVVK-601) in the 'Elektronika-100I' Computer," ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCh, 1980, issue 4, p 109, bibl. of 1 title.

1679. Lazutin, Yu. M., "Struktura matematicheskogo obespecheniya displeynoy stantsii, sopryazhennoy s EVM BESM-6" [Structure of Software for Display Station Interfaced with BESM-6 Computer], Moscow, 1978, 50 pages, (Preprint, USSR Academy of Sciences, IPM [Institute of Applied Mathematics], No 110), bibl. of 7 titles.

Describes satellite computer software, part of display system software; system consists of main BESM-6 computer that performs large volume of calculations and the small SDS-910 satellite computer connected to it by communication channel; graphic display is connected to the SDS-910. To increase display system throughput, information coming from the BESM-6 and peripherals is buffered, and requests for operation with the devices are queued. Information is reduced to form ready for use by the FORTRAN program.

8545
CSO: 1863/246

ABSTRACTS FROM 'APPLIED COMPUTER SCIENCE', NO 1, 1981

Moscow PRIKLADNAYA INFORMATIKA in Russian No 1, 1981 (signed to press 20 Feb 81) pp 229-230

METHOD OF MATRIX IDENTIFICATION AND GOAL ANALYSIS OF AUTOMATED DESIGN SYSTEMS IN INDUSTRY

[Abstract of article by V. N. Vasil'yev and V. N. Lashkov]

[Text] L. Zadeh's conceptsof fuzzy sets are used in the work to derive an algorithm for identification of elements of source information.

EVALUATION OF QUALITY OF OPERATIVE-CALENDAR PLANS

[Abstract of article by A. N. Klimov]

[Text] Method of checking plans worked out to perform construction work using production-technological models is described.

AUTOMATION OF PLANNING OF TECHNOLOGICAL PROCESS MONITORING SYSTEM USING LOCAL CRITERIA OF OPTIMIZATION

[Abstract of article by Yu. N. Shchepin and V. I. Ostrovskiy]

[Text] Problem of siting of monitoring operations in a technological process is solved according to criterion of average cost of item. The developed method achieves reduction in number of considered alternatives of monitoring plan based on local criteria of optimization.

INTERACTIVE IMAGE-TYPE DATA BASE MANAGEMENT SYSTEM (WITH EXAMPLE OF 'ACCOUNTING' PROBLEM)

[Abstract of article by V. I. Shakhmatov, A. A. Senchuk, B. A. Vorob'yev, M. Yu. Kamshitskiy, V. P. Timoshchenko and T. N. Bol'shakova]

[Text] Interactive automated system for creating and updating data bases on mini-computers is described. Example of wage calculation is given.

DESCARTES LANGUAGE AS SOURCE LANGUAGE FOR SPORA SYSTEM

[Abstract of article by I. O. Babayev, F. A. Novikov and T. I. Petrushina]

[Text] SPORA [Specialized Software for Work in Astronomy] is a software system for work in the applications program field that includes archives (data bank), a number of PPP [applications program packages], monitor and translators for the basic programming languages. The system is developed for the BESM-6 but is easily transferable to another machine.

F-LANGUAGE, FORMALISM FOR REPRESENTATION OF KNOWLEDGE IN INTELLIGENT INTERACTIVE SYSTEM

[Abstract of article by V. M. Bryabrin]

[Text] Formal language for representation of knowledge in an interactive system is described; it is based on user communication in natural language. This formalism has been implemented within the bounds of the DILOS [Dialogue Information Logical System] system at the USSR Academy of Sciences Computer Center.

PARMA, NETWORK DATA BASE MANAGEMENT SYSTEM BASED ON CODASYL RECOMMENDATIONS

[Abstract of article by V. P. Kosso, I. Ye. Kuznetsov and T. N. Sumarokova]

[Text] Described are the architecture, functional capabilities and service facilities of the network SUBD [data base management system] developed in 1976-1979 on the basis of CODASYL [Conference on Data Systems Language] recommendations (YaOD [Data Description Language] of 1973 and YaMD [Data Manipulating Language] for Cobol in 1975).

DSP DATA PROCESSING SYSTEM

[Abstract of article by V. A. Ivanov, V. A. Kondratenko and V. I. Buyanov]

[Text] Described is a data processing system oriented to data processing in an ASUP [automated production control system]. Given are basic system functions, principles of implementation as well as the system language. Some evaluations of system use are given by results of industrial operation.

VIRTUAL STORAGE ACCESS METHOD IN YES OS OPERATING SYSTEM

[Abstract of article by G. N. Staroverova and T. V. Korotayeva]

[Text] Described are data set structure, capabilities of access method and data set service program with VSAM [virtual storage access method] organization.

QUESTIONS OF COMPLEXING OF PROGRAMS IN YES OS IN ASSEMBLER AND PL/1 LANGUAGES USING MULTITASKING FACILITIES

[Abstract of article by V. M. Malyshev]

[Text] Author discusses problems arising for programmers in joining subroutines in the Assembler and PL/1 languages into a common program for execution of it in YeS OS.

IMPLEMENTATION OF APL ON YES COMPUTERS

[Abstract of article by I. I. Malashinin and A. I. Kononov]

[Text] Described is implementation of APL interactive interpreter on terminal system for YeS computers.

APPLICATION OF BINARY TREES TO ORGANIZATION OF LARGE FILES OF INFORMATION (SURVEY)

[Abstract of article by V. A. Yevstigneyev]

[Text] Author surveys foreign publications on organization of information and searching in it by using binary trees.

MUSSON MULTIMODEL DATA BASE MANAGEMENT SYSTEM

[Abstract of article by M. R. Kogalovskiy, V. V. Kogutovskiy, K. I. Makal'skiy and M. M. Vinogradov]

[Text] Authors discuss main concepts of implementation of the multilevel data base management system MUSSON [Multilevel Self-Describing General-Purpose System], its architectural features and interfaces to external level. Major concepts of the conceptual model of data and the mechanism of maintaining it are considered.

8545
CSO: 1863/229

PARMA, NETWORK DATA BASE MANAGEMENT SYSTEM BASED ON CODASYL RECOMMENDATIONS

Moscow PRIKLADNAYA INFORMATIKA in Russian No 1, 1981 (signed to press 20 Feb 81) pp 104-117

[Article by V. P. Kosso, I. Ye. Kuznetsov and T. N. Sumarokova from collection of articles "Applied Computer Science", edited by V. M. Savinkov, Izdatel'stvo "Finansy i statistika", 15,000 copies, 232 pages]

[Excerpt] Initial Specifications

The PARMA data base management system (DBMS) was developed at the NIIUMS [Scientific Research Institute of Control Computers and Systems] (in Perm') in the period 1976 to 1979. Development was based on the concepts for building data bases (BD) worked out by the Conference on Data Systems Language (CODASYL). The versions of the Data Description Language (YaOD) of 1973 [1] and the Data Manipulating Language (YaMD) for Cobol of 1975 [2] were taken as bases; Cobol and Assembler were taken as the languages included in the system. In developing the DBMS, decisions made in similar foreign systems were taken into account and additional sources were used [3]. English mnemonics were kept for all language facilities. Since CODASYL developments are rather well known [7], special terms are used below without strict definition.

The DBMS is designed for operation in the environment of YeS OS, versions 4.0 and higher with any configuration of the operating system except RSR; control tables of YeS OS declared in versions 4.0 and 4.1 are used as the monitor. The system is programmed in Assembler and has about 80,000 instructions. About 400 tracks of 5050 ZUPD or 200 tracks of 5061 ZUPD, respectively, are required for the DBMS libraries. Data bases may be created on computer systems with 256K or more of main storage.

Additional facilities are provided by an interface with the "Kama" system which provides the capability of operating in the teleprocessing mode; in this case, 512K of main storage is required.

8545
CSO: 1863/229

DSP DATA PROCESSING SYSTEM

Moscow PRIKLADNAYA INFORMATIKA in Russian No 1, 1981 (signed to press 20 Feb 81) pp 117-135

[Article by V. A. Ivanov, V. A. Kondratenko and V. I. Buyanov from collection of articles "Applied Computer Science", edited by V. M. Savinkov, Izdatel'stvo "Finansy i statistika", 15,000 copies, 232 pages]

[Excerpt] The DSP system is an applications program package oriented to programming and organization of data processing in the sphere of production control.

Purpose of the DSP System

DSP operates on YeS computers under control of the operating system (OS) and supports the full technological cycle of data processing:
creation and maintenance of a data base, including input and update, conversion of structures and types of data, monitoring of input data and changes;
search and extraction of data according to specified conditions of selection;
arithmetic and logic processing of data; and
generation of responses (documents) with rather complex structural and sophisticated design components.

The following factors were taken into consideration in developing the DSP system.

1. To one extent or another, the specific facilities for implementation of the capabilities listed above are offered by the DBMS's that are more widespread in our country. However, the advantages of a DBMS are achieved to the full extent only when the direct access storage size connected simultaneously to the computer system is sufficient to hold the entire data base. The now widespread disk storage units with a capacity of 7.25 Mbytes for medium and large systems are not very suitable.

Thus, the extensive distribution of YeS computers, on the one hand, and the lack of supporting them with direct access storage units with a large capacity (29, 100 and 200 Mbytes), on the other hand, maintain the relevance of the current use of magnetic tapes as media for storage and processing of data in computer systems based on YeS computers.

2. Maintenance of continuity of management systems created (transferred) for the YeS computers with systems on the high-volume second-generation computers--the "Minsk-32."

3. Necessity of creating simpler facilities, compared to existing ones, for man-computer communication which will allow enlisting for practical programming of ASUP [automated production control system] problems low-skilled specialists (including even non-programmers), reducing personnel training time and problem programming and debugging time.

4. Providing the capability of direct use of data stored on magnetic tapes for the "Minsk-32" computers together with data in the YeS computer formats without preliminary conversion.

8545
CSO: 1863/229

VIRTUAL STORAGE ACCESS METHOD IN YES OS OPERATING SYSTEM

Moscow PRIKLADNAYA INFORMATIKA in Russian No 1, 1981 (signed to press 20 Feb 81) pp 135-145

[Article by G. N. Staroverova and T. V. Korotayeva from collection of articles "Applied Computer Science", edited by V. M. Savinkov, Izdatel'stvo "Finansy i statistika", 15,000 copies, 232 pages]

[Text] In this article, we discuss the basic principles of operation and service capabilities of a new method of accessing data on direct access devices developed within the framework of the YeS OS operating system. This method of access, called the virtual storage access method (VSAM) is used in the virtual method of operation of the YeS OS (SVS mode) and achieves a high rate of data processing.

All earlier access methods existing in YeS OS are designed for direct access devices oriented to processing data in computers with a limited amount of main storage. Therefore, the entire data base management system, and access methods in particular, were oriented to minimizing the size of main storage required to support a specific type of access to achieve a sufficiently high rate of data access. The physical record--the smallest processing unit for a direct access device--was adopted as the unit of exchange between external (storage on a direct access device) and main storage. To access a record, the programmer used the address of the physical record on the device in the form of the cylinder and track address; therefore, for certain types of processing, the programmer was tied to the physical characteristics of the devices. The access methods developed earlier (BSAM, QSAM, QISAM, BISAM, BDAM and BRAM) are possible in YeS OS for the MFT and MVT modes and are intended for operation of computers without virtual storage.

In operating systems with virtual storage, the computer user has a considerably larger amount of storage to solve his problems; therefore, the capability of organizing a new method of access--virtual (VSAM)--has emerged. In this method, the unit of exchange between external and main storage is the control interval consisting of a certain number of physical records specified by the programmer. Physical records with a length of 512, 1024, 2048 and 4096 bytes are used in the VSAM method. Within the limits of a data set, the physical record length is fixed, which allows more efficient use of sector organization of direct access devices. The user of the VSAM access method actually has nothing to do with the physical records; he works with either the logical records or the control interval. Therefore, a new method of addressing records is used--by the relative displacement of the beginning

of a record from the first record in the data set in bytes. This method is easier for the programmer to understand and is absolutely independent of the physical characteristics of the direct access device. A four-byte value, designated an RBA [relative byte address], is intended for addressing, and this method of addressing allows access to any record in the data set. The address of the first record in a data set equals 0, the address of the second equals the length of the first logical record, etc.

The record format in a data set with VSAM organization differs from that of other access methods. The programmer in general does not specify whether his records will be fixed, variable or undefined length since the structure of the control interval allows determination of the address of the next logical record. The control interval consists of a certain number of logical records and their descriptors. Several control intervals are grouped into a control area, a larger unit of storage of a data set.

Data set records are key- or entry-sequenced. In the key-sequenced data set, the computer user can obtain access to a record by its key or by the relative byte address (RBA). If a data set is entry-sequenced, access to the record is possible only by the RBA.

A key-sequenced data set has two components--an index and data. The data component contains the logical records themselves, while the index component establishes the correspondence between the record keys and their addresses. The index has a multi-level structure. For the index component, the length of a physical record always equals the length of a control interval.

The data component and the index component (if it exists) describing it are called the structure group. The structure group for a key-sequenced data set consists of the index and data components, while that for an entry-sequenced set has only the data component.

Organization of a data set in ascending key sequence is similar to index sequential organization, while that in entry sequence is similar to sequential and direct organization.

In index sequential organization that provides access to a physical record by its key, hardware facilities in the direct access devices are used to search by record key for one greater than or equal to the key. From this stems the necessity of positioning records in sequence of ascending physical addresses on the medium. When new records are added in the middle of the data set, the later records are transferred from the tracks to an overflow area so as to not disrupt the sequence of ascending addresses.

In VSAM organization, no keys of physical records are used at all and the possibility emerges of more efficiently including new records in the data set organized in ascending key sequence. If there is free space in the control interval, the record is inserted in the space needed; if there is not enough free space for the new record, the control interval is divided in half, half of it is written into a new control interval, and the new record is inserted in the needed space. If there is no free interval in the control area, the control area is also divided in half and half of its control intervals are transferred to a new control area. In the process,

only the indexes have to be copied, establishing correspondence between the record keys and their new addresses. Thus, the overflow areas inherent to index sequential organization do not occur, and all the shortcomings of index sequential organization are eliminated.

In later versions of the VSAM access method, there will be a capability for multikey structure of a data set, when in addition to a key for the data set, additional keys may be inserted without recopying the data records, but only after building additional indexes that establish the correspondence between the new keys and addresses of the records.

The VSAM method provides for both sequential and direct processing of records, and also implements all the capabilities offered by the other access methods for direct access devices.

For data sets organized in ascending key sequence, VSAM allows obtaining direct access to a record by its key or address, adding a new record, erasing a record and changing record length.

For data sets organized in entry sequence, each new record is added at the end and insertion of records in the middle, changing record length and obliteration of records are not permitted.

The VSAM method allows sequential, partially sequential or the direct type of data processing, and at the same time combining the different types of data set processing is permitted. In sequential or partially sequential processing, the programmer may specify the beginning of processing in the form of a key or address of a record, and the key may be incomplete. Partially sequential processing is used for a data set organized in ascending key sequence when not all records in the data set have to be processed, but the required records can be requested in ascending order of their keys and retrieval at the index of the lower level, beginning with the key of the preceding record, can be performed faster than direct retrieval of the key beginning at the index of the higher level.

Sequential processing of a data set may be performed by key and address.

The VSAM method in operating with data sets supports one language of access--macroinstructions for building control blocks and obtaining from them information and macroinstructions for processing requests.

The macroinstructions for building control blocks are:

ACB, used to build the access control block during program translation;

EXLST, designed to build an exit list for user programs during program translation. One can specify an exit to a program for processing logical input-output [IO] errors, a program for processing IO errors associated with equipment functioning, a program for checking access permission and a program for recording events;

RPL, which organizes the request parameter list during program translation;

GENCB, which builds the request control block, a list of exits or a list of request parameters during program execution;

MODCB, used to modify the access control block, exit list or request parameter list during program execution;

SHOWCB, used to read the fields of the access control block, exit list or request parameter list during program execution; and

TESTCB, designed to test the fields of the access control block, exit list or request parameter list during program execution.

Request processing macroinstructions are used to write, read or erase data records in the data set. The main macroinstructions in this group are:

GET -- for reading data records;

PUT -- for writing data;

ERASE -- for erasing data records;

POINT -- for setting at the record needed;

CHECK -- for suspending execution of processing before completion of request; and

ENDREQ -- for ending the request.

To establish a link between the processing program and a specific data set, the programmer has to build in his program an access control block (ACB) which in the VSAM is used instead of the data control block (DCB) used by other access methods. In the access control block, the name of the DD statement of the job control language is assigned, and in the DD statement when the job is started, the data set name that will be processed is indicated. Before starting processing of a data set, the ACB must be opened using the macroinstruction OPEN and closed at the end of processing by using the macroinstruction CLOSE.

Access to indexes in a data set organized by ascending keys is performed in one of two ways:
the structure group is opened and the macroinstructions GETIX (read index) and PUTIX (write index) are used;
one index component is opened and the macroinstructions for ordinary data processing are used (GET, PUT, etc.).

Processing of one index component is identical to processing of a data set in entry sequence. The index component itself has no indexes and therefore may not be processed by the access method by key.

The user may not duplicate or replace the index processing performed by VSAM during ordinary processing of data records.

To provide for standard operation with data sets, instead of a large number of utility programs used for operating with data sets of a different organization, VSAM has a multifunction utility program (AMS) which has its own function request language.

AMS defines VSAM data sets, loads records into them, converts sequential or index sequential data sets into VSAM format, lists the VSAM catalog or data records, copies data sets, creates copies of catalog and data set, facilitates data set portability between operating systems and erases data sets.

The programmer requests the desired function of the multifunction utility program using instructions of this program and their parameters. The AMS program instructions are divided into two types: functional and modular.

The main functional instructions that define the operation to actually be performed are:

DEFINE -- for organization of VSAM catalogs, catalog elements describing a data set, and zones of external storage on direct access devices that will be used by the VSAM access method to allocate storage from them for data sets;

DELETE -- for deleting elements in a catalog;

ALTER -- for changing elements in a catalog;

LISTCAT -- for listing catalog elements;

PRINT -- for printing a VSAM data set, as well as index sequential data sets;

REPRO -- for copying data sets, converting sequential and index sequential data sets to VSAM format, converting a VSAM data set and index sequential data set to a sequential data set, organization of backup copy of catalog and restoration of the catalog from the copy;

VERIFY -- used in detection of an error when the data set is closed to guarantee the correctness of elements of the catalog that describes the data set;

CHKLIST -- defines the volume of magnetic tape for the checkpoint program;

EXPORT -- for creating a backup copy of a VSAM data set and preparing a VSAM data set and user catalogs for use in another operating system; and

IMPORT -- for restoring a VSAM data set from its copy and for providing the capability of using in another operating system the VSAM data sets and user catalogs prepared using the EXPORT instruction.

The functional instructions DEFINE, ALTER, DELETE, LISTCAT, REPRO and PRINT may be used for processing data sets with an organization differing from VSAM.

The modular instructions allow creating conditions for execution of functional programs. The main instructions of this type are:

sequence of instructions IF ... THEN ... ELSE which check the completion code of the preceding functional instruction and as a function of its value bypass or include execution of the sequence of functional commands;

SET -- changes the completion code; and

DO ... END defines the start and end of a sequence of instructions.

The completion code is set when each functional instruction is executed and has the following value:

0 -- complete execution of the instruction;

4 -- instruction was executed, but a warning was made about the peculiarity of the execution; for example, the required element was not found in printing a catalog element;

8 -- instruction was executed, but some of the requested functions were not executed, for example, in deleting a catalog element, the required element was not found;

12 -- instruction not executed due to logic error; and

16 -- error in execution, which causes a halt in execution of the entire stream of instructions. This code may be set, for example, when the system could not open a data set for system output or a permanent IO error was detected in system output.

In the IF instruction, one can check the actual completion code of the last instruction, designated LASTCC, or the maximum permitted value of the completion code, designated MAXCC.

An example of the use of the instruction sequence IF ... THEN ... ELSE is:

```
IF MAXCC < 8
 THEN
    DO
    instruction sequence
    END
 ELSE
    DO
    instruction sequence
    END
```

The multifunction utility program may be executed as a job or job step, and also may be called from another program.

Each data set that is to be processed by the VSAM access method must be defined in a catalog by using the multifunction utility program, i.e. catalog elements must be created for it that describe the occupied zone, name and other characteristics.

In the VSAM access method, catalogs are used as the basic source of information on VSAM data sets and the volumes on which they are located. There are two types of catalogs--master and the user catalog. A master catalog is required for VSAM; user catalogs are optional. Each VSAM catalog contains records defining it. System catalog contents point to master catalog contents, and the contents of the latter to user catalogs.

A catalog includes information on the space occupied by a data set, indicates data set access restrictions, contains usage statistics and links the RBA byte address to physical addresses.

All data sets located on a volume must be cataloged in one master or user catalog. A data set is defined only in one catalog.

The master catalog may not be located on the main resident volume for the system since if it is damaged it would not be possible to restore it.

If a VSAM data set is cataloged in a user catalog, this catalog must be assigned by the statement DD JOBCAT or STEPCAT. The STEPCAT statement defines the catalog used in executing this job step, and the JOBCAT statement in executing the whole job. If several catalogs are needed, they are described as ordinary concatenated data sets in YeS [Unified Series] OS.

The use of user catalogs raises the reliability of system operation. Catalogs may be located on each volume instead of one common catalog. This reduces the number of accesses to a given catalog to the minimum and if some catalog is destroyed the others are still available.

When a catalog is created, the programmer may specify a special facility that makes it possible to restore it. In this case, a special area is allocated on a volume for records that contain the information required to restore the catalog when there is a system or equipment malfunction.

VSAM obtains storage on a volume and manages it in units called zones. A zone occupies a certain number of tracks and cylinders. One zone may have one or more data sets, and in turn a data set may occupy several zones. Zones are described in a VSAM catalog.

A catalog manages the volumes containing VSAM data set defined in this catalog. All VSAM structural groups and all zones on a volume must be cataloged in the catalog that manages the volume.

A tag is set in a volume table of contents to indicate that this volume is managed by a VSAM catalog. The latter contains the volume entries for each volume that it owns. Each entry describes the characteristics of the volume, all zones and objects of VSAM that use storage on the volume. Only one VSAM catalog may own a volume.

Volume management using a VSAM catalog does not affect data sets on this volume that have an organization different from VSAM. These data sets may be located on a volume managed by a VSAM catalog and may be cataloged not in this catalog. An OS catalog can be considered an ordinary data set and it may be located on a volume managed by a VSAM catalog.

To free a volume from management by a VSAM catalog, one has to get rid of all VSAM objects on the given volume by using the multifunction utility program. Then the DELETE SPACE instruction of the multifunction utility program is used to delete all zones and remove the entries of this volume from the catalog and remove the tag of ownership of this volume from the table of contents.

A user may process a VSAM catalog as a data set or in a special way. To process a catalog as a data set, a user has to link his processing program with this catalog and use the VSAM macroinstructions (GET, PUT, etc.) for processing. To obtain access by the method special for the catalog, a programmer has to use the special multifunction utility program for data sets (AMS). Thanks to the use of the multifunction utility program for processing a catalog, the capability of performing the following functions is available:

creating a catalog (instructions DEFINE MASTERCATALOG and DEFINE USERCATALOG);

including records in a catalog (DEFINE);

changing records in a catalog (ALTER);

deleting records in a catalog (DELETE); and

listing the records of a catalog (LISTCAT).

Catalog information pertaining to a specific data set may be obtained by a processing program by using the macroinstruction SHOWCAT irrespective of whether or not the given data set is opened. Information is issued from the catalog specified by the DD statements with the names STEPCAT or JOBCAT, or from several catalogs if the given DD statements define concatenated data sets, or from the master catalog if the JOBCAT or STEPCAT statement is missing or a catalog description is not found in them.

To obtain access to a catalog as to a data set, it has to be described by a DD statement of the job control language (not JOBCAT or STEPCAT), an access control block (ACB) has to be built for it in the program and it has to be opened by the macroinstruction OPEN. Catalog records may be accessed by key or address. In this operation, it is recommended that catalog records only be read and updated. Since catalog records have a complex interrelationship, including new records or deleting records from a catalog is not recommended. To add or delete catalog records, it is recommended that the multifunction utility program be used.

VSAM affords a greater capability for restricting access to data sets compared to other access methods.

A programmer may restrict access to structural groups, data or index components, VSAM catalogs and catalog entries for data sets. The capability exists to specify the following levels of data access restrictions:

restriction of any access which requires confirmation of permission for execution of all operations (reading, updating, adding and deleting) with a data set, any index and records of the catalog that correspond to this data set;

restriction of access by control interval;

restriction of access for updating. This allows reading, updating, adding or deleting of records in a data set only with confirmation of permission for execution of these operations; and

restriction of access for reading. This allows reading of data and catalog records with confirmation of permission, but does not allow adding, changing or deleting them.

To restrict access to a data set, one must also restrict access to the catalog in which the given data set is defined. Each higher level of access restriction allows execution of a lower level operation.

When YeS OS requests permission for access to a data set from the computer operator, a code can be assigned to be used instead of the data set name, so that the computer operator will not know the data set names to which access is restricted. One can specify the number of attempts which a computer operator may use for permission of access to a data set.

In addition to the standard check for permission to access a data set, VSAM allows a programmer to use an additional check for access permission by using a user program. This program is called after YeS OS obtains permission for the given type of access.

The VSAM access method permits a programmer to specify permission to access a data set in his program which frees him from a message to the computer operator with

information on his data sets. VSAM requests access permission from the computer operator only when access permission is not specified in the program.

A great advantage of the VSAM access method is the simplicity of its use. The programmer has to be familiar with just several macroinstructions and has to indicate a minimum of information on the data set in the program and job control language (substantially less than for other YeS OS access methods). For example, in the DD statement for VSAM, one has to specify only the set name (DSNAME) and its disposition (DISP). The programmer does not have to describe record formats or know the physical organization of the data set; his programs are device independent and he need not be concerned about blocking of records and bufferization. This reduces outlays for programming and the possibility of errors.

The VSAM method exceeds the capabilities of all YeS OS access methods used for direct access devices in providing faster access to data. Therefore, in systems with virtual modes of operation, it will replace all other access methods except the library. Since many programs have now been developed with index sequential organization of data sets, the VSAM access method includes programs called ISAM interface with VSAM that permits a program written by using the ISAM access method in fact to process a data set with VSAM organization. All that is required are minor changes in the job control language statements and a description of the data sets by using the multifunction utility program.

If there is a good algorithm of randomization for a data set with direct organization, the use of BDAM may be retained.

* * *

Since the experience of operating with other access methods was taken into account when the VSAM access method was developed, certain shortcomings and inconveniences of the other access methods were eliminated in the design of VSAM. The process of writing reenterable programs carrying on processing of data was made easier; the capability of different simultaneous processing of data sets was introduced, which eliminates the need of repeatedly opening and closing a data set when switching from one form of processing to another, for example from direct processing to sequential; new capabilities for protecting data sets were introduced; and access to information on access restriction is not authorized for even the systems programmer and computer operator, but only for the programmer himself.

Therefore, the VSAM access method should find extensive application in data processing systems using the YeS OS operating system in the virtual mode of operation.

8545
CSO: 1863/229

IMPLEMENTATION OF APL ON UNIFIED SYSTEM OF COMPUTERS

Moscow PRIKLADNAYA INFORMATIKA in Russian No 1, 1981 (signed to press 20 Feb 81) pp 155-170

[Article by I. I. Malashinin and A. I. Kononov from collection of articles "Applied Computer Science", edited by V. M. Savinkov, Izdatel'stvo "Finansy i statistika", 15,000 copies, 232 pages]

[Text] The extensive attraction to using computers of specialists in various fields of knowledge has required offering them convenient and on-line facilities for man-machine communication. This pertains to both language and the form of intercourse. In this article, we discuss questions of implementing a terminal system with the APL language which has become widespread abroad and which affords direct user access to allocated computer resources through terminals and software facilities. The simplicity of the language syntax, brevity and precision of expressions, powerful facilities for operating with multidimensional arrays (vectors, matrices, etc.), and the capability of on-line computations and creation of programs (functions) make APL attractive for a broad range of users [1, 7].

This article focuses on APL notation and use of terminals. Also covered are the principles of building the systems and operating it under the control of YeS OS.

System Concepts

The APL system implements time sharing among a group of remote or local terminals. The terminal is an electric typewriter or console typewriter with a control unit, such as for example in APL \360 [5].

In the APL system, a user exchanges information with the system only through a user terminal. All statements entered by the user from a terminal are divided into system commands and APL statements. All information output to the user consists of system messages, computation results and program texts [4, 5].

Time sharing is implemented by the system supervisor, and the APL interpreter identifies and processes information coming in from the terminals.

The system operates under the control of YeS OS of the MFT or MVT type and takes up one partition or zone of main storage. The APL system functions the following way. Upon connection to the system, the user is allocated workspace of fixed size defined at system generation. This space can be named and kept in the user's

personal library. It can subsequently be called into the active zone for further work. The active zone includes part of main storage and storage on direct access devices used for the swapping [svoping] process. In addition to personal libraries, the capability exists to create common libraries accessible to all system users.

Storage extents on disk units designed in the standard of sets of Yes OS are used to store user libraries and organize swapping of active work areas.

By using a special utility, there is the capability of creating transportable copies on magnetic tape of individual work areas and libraries for storage or transferring them to other installations.

Users in the APL system work independently of each other while under supervision of the APL operator. The latter, in addition to the usual user capabilities in the system, is given privileged facilities to control the system.

The APL system provides an adequate level of security and reliability.

The APL Language

APL (A Programming Language), developed by K. E. Iverson [3], is designed for engineers and scientific and technical calculations although its field of activity is not limited to these areas. The language is unconventional for professional computer users, but easily mastered by nonprofessional computer users since it is almost devoid of those auxiliary details that any user has to know to solve problems on computers [7, 8].*

The wealth of language symbols minimizes terminal input to execute particular operations. APL offers more than 40 built-in functions which substantially eases programming.

An important place in this language is allocated to facilities for working with vectors and matrices of numbers and symbols. There are facilities for operating with text constants.

The mode of immediate interpretation allows the user to use the terminal as a calculator, and the program definition mode allows input, correction and reproduction of a program text.

In executing programs (functions), APL allows following the course of their execution and stopping it.

In the APL system, there is the capability of writing interactive problems. The main set of functions offered by APL are called primitive functions. Based on them, it is possible to create any number of additional functions; thus, the language is easily expanded.

---

* Shirokov, F. V., "Basic Concepts of APL," in the book "Algoritmy i organizatsiya resheniya ekonomicheskikh zadach" [Algorithms and Organization for Solving Economic Problems], Moscow, Statistika, issue 14, 1980.

A distinguishing feature of APL is the simplicity of data representation and the absence of specifications in defining a variable, since the value assigned to it characterizes the variable. In addition, there is no traditional prioritization of operations in APL expressions—all operations are of equal priority, and an expression is computed sequentially from right to left. The number of messages to the user has been held to a minimum and does not exceed 40.

The APL Language for the Unified System of Computers

The main difficulties in using APL for the YeS computers are the lack of a domestic terminal with the proper keyboard and means of communication. Therefore, in using APL on domestic equipment, there arose first of all the problem of changing its symbolic notation.

The symbol of the most widespread version of APL is described in Backus normal form the following way [6]:

APL symbol : : = letter ❙ number
❙ special symbol ❙ blank ❙ '

letter : : = A|B|C|D|E|F|G|H|I|J|K| L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|
A|B|C|D|E|F|G|H|I|J|K| L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|
number : : = 0|1|2|3|4|5|6|7|8|9

special symbol ::= ¨|¯|<|≤|=|≥|>|≠|∨|∧|−|÷|+|×|
?|⍵|∊|⍴|~|↑|↓|⍳|○|⋆|→|←|⍺|⌈|⌊|_|∇|∆|∘|⎕|(|)|[|]|⊂|⊃|
∩|∪|⊥|⊤|∣|;|:|\|,|/|⍱|⍲|⍒|⍋|⍉|⌽|⍟|⊖|⌹|⍞|!|⌶|⍀|⌿|⍫|⍝
blank : : = vacant position

Thus, the APL alphabet consists of 131 symbols.

All APL symbols are divided into simple and combined (ligatures). Simple symbols are input from the APL terminal by one press of the keys, combined--by printing a simple symbol over another.

When input from the terminal keyboard, each APL symbol is converted into a single value in the internal representation of the APL system, from which it is recoded into the external representation only when output to the terminal. The APL terminal keyboard contains 88 simple characters (table 1).

The majority of the special symbols have a specific functional purpose. The symbols ⊂ ⊃ ∩ ∪ ¨ ⍵ ⍺ do not define internal functions of the APL interpreter and are intended for subsequent implementations of APL. The illegal symbol obtained by a combination of the characters O, U and T has a special meaning.

The combination symbols include: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

⍱ ⍲ ⍒ ⍋ ⍉ ⌽ ⍟ ⊖ ⌹ ⍞ ! ⌶ ⍀ ⌿ ⍫ ⍝ , input of which is effected by entering the first simple symbol, backspacing and entering the second simple symbol (table 2).

Table 1. Correspondence between Simple Symbols of APL Language and Its Modification for Unified System of Computers and YeS-7077 Type Terminals

APL Symbol	Symbol Name	DKOI Replacement Symbol	Main Use of Symbol in APL Expressions
A - Z	letters of alphabet	A - Z	identifiers of work areas, functions (programs) and variables
0 - 9	numbers	0 - 9	numeric values
..	double period	none	symbol
¯	superscript	¬	sign of negative number
<	less than	<	operation of checking for correspondence to "less than"
≤	less than or equal	none	operation of checking for correspondence to "less than or equal"
=	equal	=	operation of checking for correspondence to equality
≥	greater than or equal	Б [B]	operation of checking for correspondence to "greater than or equal"
>	greater than	>	operation of checking for correspondence to "greater than"
≠	not equal	Ж [Zh]	operation of checking for correspondence to inequality
∨	logical OR	Л [L]	logical "OR" operation
∧	logical AND	И [I]	logical "AND" operation
-	minus	-	subtraction operation
÷	divide	%	division operation
+	plus	+	addition operation
×	multiply	#	multiplication operation
?	question	?	generation of random numbers
ω	omega	none	symbol
ε	epsilon	Э [E]	determination of coincident elements of two vectors or matrices
ρ	rho	Ь [']	operations with dimensionality of vectors and matrices
~	tilde	Ц [Ts]	logical "NOT" operation
↑	up arrow	Ю [Yu]	selection of vector elements

APL Symbol	Symbol Name	DKOI Replacement Symbol	Main Use of Symbol in APL Expressions
↓	down arrow	none	dropping of vector elements
⍳	iota	И [Y]	generation of sequence of integers, determining coordinates of elements of one vector in another
○	circle	@	trigonometric functions and operations with pi
*	asterisk	*	raising to power operation
→	right arrow	Ы [Y]	jump to marker or exit from program
←	left arrow	З [Z]	assignment of value to variable
⍺	alpha	none	symbol
⌈	maximum	Г [G]	finding largest of two quantities
⌊	minimum	Ч [Ch]	finding smallest of two quantities
_	underline	_	see table 2
∇	nabla	Ъ [U]	beginning and ending of function (program) definition
Δ	delta	Д [D]	see table 2, tracing of program
∘	jot	¤	outer product of matrices
'	apostrophe	'	definition of character string
⎕	quad	П [P]	request for data input into program
(	left parenthesis	(	changing order of computation in APL expression
)	right parenthesis	)	
[	left bracket	[	enclosure of subscripts (coordinates) of elements of a vector or matrice
]	right bracket	]	
⊂	subset	none	symbol
⊃	inclusive set	none	symbol
∩	intersection	Я [Ya]	symbol
∪	union	none	symbol
⊥	base	Ш [Sh]	recoding of vector into scalar
⊤	top	Щ [Shch]	recoding of scalar into vector

APL Symbol	Symbol Name	DKOI Replacement Symbol	Main Use of Symbol in APL Expressions
|	stile	!	absolute value or finding residue (of a number modulo another)
;	semicolon	;	index separator; mixed output separator
:	colon	:	label separator in program
\	backward slash	Ф [F]	"expansion" of matrix across last coordinate
/	slash	/	"compression" of matrix across last coordinate
,	comma	,	"merging" of matrices, raveling of matrix into vector
.	period	.	separator of whole and fractional part of a number, generalized product of matrices

When a combined character is entered, actually three code combinations are sent to the channel: the first symbol, the backspace and the second symbol. The same occurs when a combined symbol is output to the terminal.

In addition to the codes for the set of simple APL symbols and the [BS] (backspace) code, the APL interpreter distinguishes the code combinations between the special keys [LF] (line feed) and [CR] (carriage return) which are used in correcting text and completing entry of information from the APL terminal.

The majority of YeS computer terminals based on the "Konsul-260.1" typewriter contain the following set of DKOI symbols [2]:

The Latin alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ;

the Russian letters that differ from the Latin:

БГДЖЗИЙЛПУФЬЦЧШЩЭЮЯ;

[B G D Zh Z I Y L P U F ' Ts Ch Sh Shch E Yu Ya];

the numbers:

0 1 2 3 4 5 6 7 8 9;

and the special symbols:

+ − # % * ! ? ¤ & ' ( ) / = : ; [ ] < > ⌝ - @ , . space

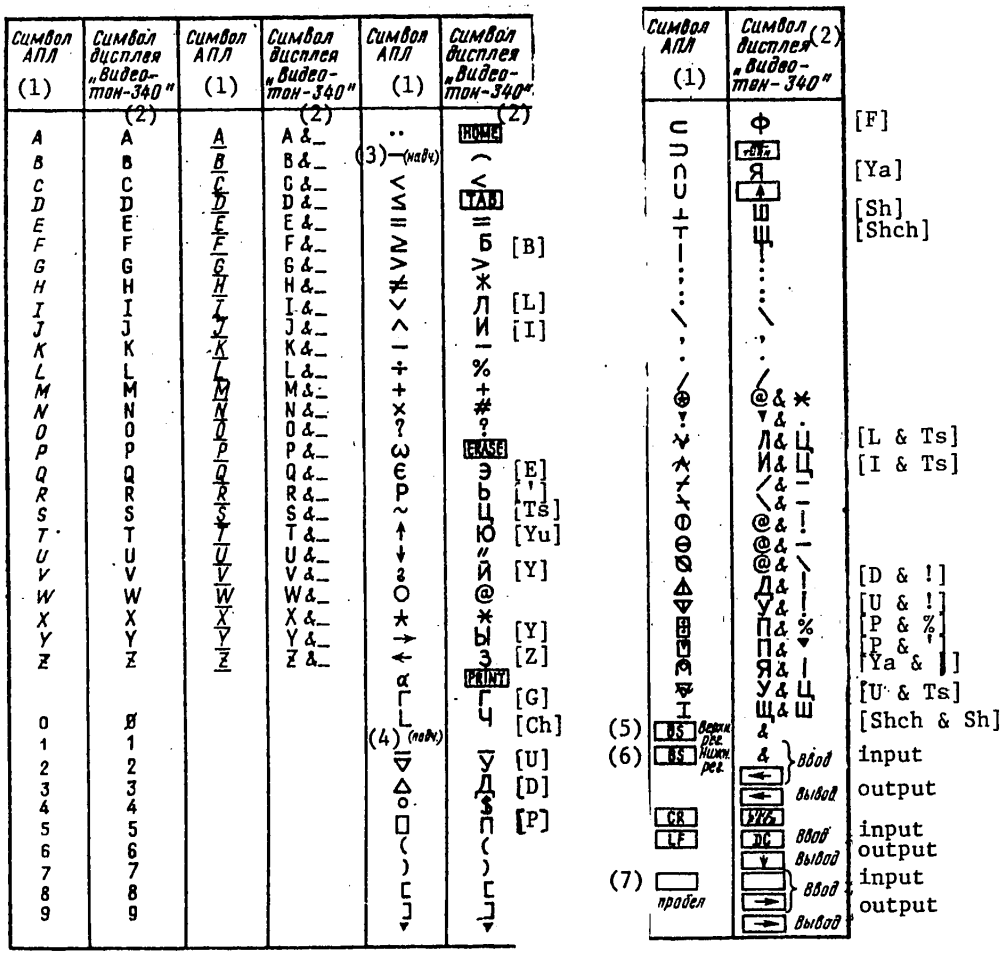
Based on this set of symbols, the notation for the YeS APL language was built for the YeS-7077 type terminal.

Table 2. Correspondence between Combined Symbols of APL Language and Its Modification for Unified System of Computers and YeS-7077 Type Terminals

Символы АПЛ (5)		Символы модифицированного языка АПЛ (6)		Main Use of Symbol in APL Language
Составляющие символы (1)	Комбинированные символы	Комбинированные символы при выводе	Составляющие символы при вводе (4)	
1	2 (2)	3 (3)	4	5
O *	⊛	⊛	@ & *	logarithm function
! .	!	!	! & .	factorial function, combinations function
∨ ~	⍲	Ч	Л & Ц	[L & Ts] logical NOR operation
∧ ~	⍱	Ц	И & Ц	[I & Ts] logical NAND operation
/ –	⌿	⌿	/ & –	"compression" across first coordinate
\ –	⍀	ф	Ф & –	[F & –] "expansion" of matrix across first coordinate
O |	⌽	@	@ & !	rearrangement of matrix elements across first coordinate
O –	⊖	@	@ & –	rearrangement of matrix elements across last coordinate
O \	⍉	@	@ & Ф	[@ & F] matrix transposition
Δ |	⍋	Д	Д & !	[D & !] ascending sort
∇ |	⍒	У	У & !	[U & !] descending sort
⍞ ÷	⌹	П	П & %	[P & %] matrix inversion, solving a system of linear equations
⍞ '	⍞	П	П & '	[P & '] literal input into program
∩ °	⍝	Я	Я & ¤	[Ya & ¤] comment line in program
∇ ~	⍣	Ц	У & Ц	[U & Ts] lock to inhibit printout of program text
T ⊥	I	Щ	Щ & Ш	[Shch & Sh] system dependent functions
	Подч. алфавит:	Подчер. алфавит:		underscored alphabet
A _	A	A	A & _	identifiers of variables
B _	B	B	B & _	
.....	.....	.....	.....	
Z _	Z	Z	Z & _	
	Спец. символ	Спец. символ		special symbol
OUT	⍬		O & U & T	symbol for rejection of literal input into program

Key:
1. Component symbols
2. Combined symbols
3. Combined symbols when output
4. Component symbols for input
5. APL symbols
6. Symbols of modified APL language

Table 3. Correspondence between APL Symbols and Symbols on the Keyboard of the "Videoton-340" Display



Символ АПЛ (1)	Символ дисплея „Видеотон-340" (2)	Символ АПЛ (1)	Символ дисплея „Видеотон-340" (2)	Символ АПЛ (1)	Символ дисплея „Видеотон-340" (2)
A	A	A	A & _	⍤ (3) (навч)	HOME
B	B	B	B & _	(	(
C	C	C	C & _	<	<
D	D	D	D & _	≤	TAB
E	E	E	E & _	=	=
F	F	F	F & _	≥	Б [B]
G	G	G	G & _	>	>
H	H	H	H & _	≠	*
I	I	I	I & _	∨	Л [L]
J	J	J	J & _	∧	И [I]
K	K	K	K & _	−	−
L	L	L	L & _	+	%
M	M	M	M & _	÷	+
N	N	N	N & _	×	#
O	O	O	O & _	?	?
P	P	P	P & _	⍵	ERASE
Q	Q	Q	Q & _	∈	Э [E]
R	R	R	R & _	⍴	Ь [']
S	S	S	S & _	~	Ц [Ts]
T	T	T	T & _	↑	Ю [Yu]
U	U	U	U & _	↓	"
V	V	V	V & _	⍳	Й [Y]
W	W	W	W & _	○	@
X	X	X	X & _	*	*
Y	Y	Y	Y & _	→	Ы [Y]
Z	Z	Z	Z & _	←	З [Z]
				⍺	PRINT
				⌊	Г [G]
0	Ø			⌈	Ч [Ch]
1	1			(4) (подч) _	
2	2			∇	У [U]
3	3			△	Д [D]
4	4			⍬	$ [P]
5	5			□	П [P]
6	6			(	(
7	7			)	)
8	8			⌋	Ъ
9	9			⌊	▼

Символ АПЛ (1)	Символ дисплея „Видеотон-340" (2)	
⊂	Ф	[F]
⊃	ПУСК	
∩	Я	[Ya]
∪	▲	
+	Ш	[Sh]
⊥	Щ	[Shch]
|	|	
…	…	
/	/	
⌿	'	
.	.	
⍀	/	
⍉	@ & *	
!	▼ &	
⍲	Л & .	
⍱	И & Ц	[L & Ts]
⍝	/ & _	[I & Ts]
⍒	@ & !	
⍋	@ & -	
⌽	@ & /	
⊖	Д & !	[D & !]
⍟	У & !	[U & !]
⍕	П & %	[P & %]
⍎	П & '	[P & ']
⍪	Я & |	[Ya & |]
⍙	У & Ц	[U & Ts]
⍠	Щ & Ш	[Shch & Sh]
⍞	&	
(5) BS верхн. рег.	& } ввод	input
(6) BS нижн. рег.	& } вывод	output
←	←	
CR	ПУСК	input
LF	DC ввод	output
↓	↓ вывод	
(7) пробел	ввод	input
→	→	output
	вывод	

Key:

1. APL Symbol
2. "Videoton-340" display symbol
3. superscript
4. underline
5. uppercase
6. lowercase
7. space

In doing so, these were the guidelines followed:

keep the visual perception of the modified language program close to that of the APL program;

wherever possible, give a meaningful value to the special symbols;

keep the combined symbols in the language;

expand the language with the capability of using the Cyrillic alphabet; and

keep the continuity of the libraries of programs written at existing installations in the standard APL language.

Table 1 shows the correspondence of the simple APL symbols to the symbols in the modified language; table 2 shows the correspondence between the combined symbols and their component elements.

From these tables it is evident that no change was made to the majoirty of symbols coincident in graphic design. The YeS APL language contains all the Russian letters, which allows use of them in literal constants and comments, but in other cases they have a special meaning. Thus, the letter П [P] ("pechatayte" [type!]) is used to request data input into the program; bI [Y] ("vyyti" [to exit]) is used to jump to the marker or exit the program; З [Z] ("zanesti" [to make an entry]) is used to assign values to variables; and И, Л and Ц [I, L and Ts] are used to define the logical operations AND, OR and NOT ("otritsaniye" [negation]) [i = and; ili = or].

The combined symbols in the language are implemented the following way: On input, one types the triads of symbols of the first simple symbol, the symbol & instead of the backspace and the second simple symbol; on output, the symbol & is replaced by the backspace code and the actual image of the combined symbol is printed out. The correspondence between the APL combined symbols and those used in the YeS APL modified language is shown in table 2. To generate the carriage return code (since no code combination is generated when the carriage return key is pressed on the "Konsul-260.1"), the key is used which does not cause printing of the symbol, but sends the corresponding symbol code to the computer [2].

The APL symbols $\cdot\cdot\ \omega\ \downarrow\ \alpha\ \subset\ \supset\ \cup\ \leqslant$ have no equivalent in the modified language, but since they are rarely used, their absence has almost no effect on the capabilities of the language. When a program written at another installation is printed out, these symbols are replaced by blanks.

The modification, discussed above, to the APL notation was also used in implementing the display version of the terminal system. In doing so, the restrictions mentioned were eliminated and the language was given a number of important additional facilities for communication with a terminal. In this case, the terminal used can be either the YeS-7066 display from the YeS-7906 complex, connected to the multiplexer channel, or the "Videoton-340" display.

The "Videoton-340" as an APL Terminal

Interest in using the "Videoton-340" display as an APL terminal arose due to a number of factors: high output rate, no mechanical elements, high quality, broad

capabilities in text editing and input-output management, standard set of Latin and Russian characters and special symbols, large information capacity of the screen, and the capability of printing information from the screen on a special alphameric device. In addition, the technical capabilities of the display allow using it at a considerable distance away from the computer without special software or hardware for communication [9].

The main obstacle to using the "Videoton-340" displays (or the "Videoton-YeS-7168") with YeS computers beginning with the model YeS-1020 and up is the impossibility of connecting them directly to the computer channel. This required development of a special group switch that allows connection of up to 48 displays to the YeS computer multiplexer channel. In doing so, the instruction set corresponding to the YeS-7077 console was selected as the most suitable; this afforded continuity with the APL implementation for the YeS-7077 and compatibility with the subset of instructions for the YeS-7906.

The group switch is the control unit for the displays connected to it; it also converts information coming in from DKOI code into KOI-7 code for the "Videoton-340" unit and vice versa.

Since the APL system does not use a YeS OS access method, terminals engaged with the APL system are not distributed by the OS scheduler and do not require description when OS is generated. However, it makes sense to describe them anyway as devices of the YeS-7077 or YeS-7066 type with addresses of the multiplexer channel from the interval of 050 to 07F.

Such substitution allows employing software, which uses the subset of instructions for the YeS-7077 and YeS-7066 devices, for operation with the "Videoton" displays. Thus, when a "Videoton" is described as a YeS-7077, it can be defined as a YeS computer operator console, and if described as a YeS-7066, it can be used as a terminal for the interactive systems "Kama," SRV [time-sharing system], SPOK [system of programming of teaching courses] and certain others.

The "Videoton" display keyboard connected through the group switch allows transmission of 99 code combinations. These are the codes of the 26 Latin letters, the 20 Russian letters drawn differently from the Latin, the 10 numbers, 28 special symbols including the space, and the 15 control symbols for the terminal.

The APL language notation for YeS-7077 type terminals, shown in table 1, was revised to account for the capabilities of the "Videoton" terminal and became that shown in table 3. The modified version of the language was given the working name of APLVIDEO.

Since it is not possible physically to display the combined characters on the screen, they are represented in APLVIDEO by triads of symbols both on input and output.

Some of the APL symbols were replaced by the symbols (code combinations) of the terminal control keys that are shown in table 3 as rectangles with the corresponding letters.

For convenience of operation from the terminal, a number of the APL control symbols are interpreted differently on input and output to the terminal. Thus, the symbol

[BS] for the lower case of the prototype terminal at input is represented by the character & or the [←] of the "Videoton"; when this symbol is output, the backspace situation is simulated: the cursor is shifted to the left one position which is equivalent to using the [←] key.

The symbol "blank" corresponds to pressing the special key [ ] on the display keyboard. It is shown as a space on the screen. However, when correcting function definitions, the [→] key must be used; pressing this key generates the "blank" code too, but in doing so the symbols above the cursor are not replaced by spaces: the entire line being corrected remains visible while the cursor is moved below it to the position to be corrected.

Features of User Operation from the "Videoton-340" Terminal in the APLVIDEO System

The user operating procedure from the "Videoton" terminal is somewhat different from user operation with a special APL terminal.

The display is ready for operation in the system when it is in the ONLINE status. To put the display in operation, the keys [POWER] and [ON] [LINE] are used.

Information input operations from the display depend on whether a given terminal is a user terminal or an APL operator terminal (when only one terminal is connected in the system, it is automatically considered the APL operator terminal). A user begins text input directly at the place indicated by the cursor. But the operator has to first send an interrupt to the system using the key [RETURN] and only after this can he begin input of text. In both cases, information input has to be ended by sequentially pressing the keys [LINE] [FEED] [ETX].

This mode of terminal input can be called direct input. In the process, each symbol that appears on the screen is sent through buses to the computer channel. Thus, it is not possible to correct input text by replacing and inserting. However, APL offers its own capabilities for correction of text just input if the input end signals [LINE] [FEED] [ETX] have not yet been sent to the system. For the display, this capability is used as follows: When an error in text is detected, the cursor is moved by the [←] key to the first incorrect symbol from the left; it is then erased by the [DC] key; then the corrected remainder of the text is again input from this position.

In addition to direct input, using the "Videoton" display as a terminal for an APL system allows composing text in the offline mode. For this, the display is placed OFFLINE by pressing the appropriate key. Then text may be composed starting from any line. In the process, one can make use of all the offline editing facilities by using the keys [←] [→] [↑] [↓] [ERASE] [DL] [IL] [DC] [IC], and the symbols corresponding to them are not considered part of the text. Text composition is ended by the combination [LINE] [FEED] [ETX]; then the cursor is set under the first line of text and the [SEND] key is pressed. This operation causes sequential reading of the composed text and sending it into computer storage; in doing so, the display is put into the ONLINE state. When the symbols [LINE] [FEED] are encountered in the composed

text, the first such symbol in the line, reading from the left, is considered the end of input from the given line.

To correct programs by APL facilities using the "Videoton," the keys [→] and [↓] or [DC] are used. The cursor is moved by the [→] key to the position to be corrected for insertion of the necessary symbols in the line. Pressing the [DC] key right after the number of the program line to be corrected causes this line to be removed from the program text and the line numbers to be renumbered.

In correcting lines containing combined symbols, remember that they occupy three positions on the screen, but just one position as a language symbol; consequently, in moving the cursor, it is necessary to bear in mind the actual number of APL symbols passed over by the cursor.

Output to the screen from the system starts, as a rule, at the position where the cursor is set. In doing so, the amount of output may exceed the space remaining to the end of the screen, and also in general may exceed the size of the whole screen. In this case, after the last line of the screen is filled, output continues on the first line, wiping out the information already on the screen.

The following facilities are available to prevent this. First, the [ROLL] key can be used to shift the text up a line when the last line on the screen is filled. In the process, the very top line on the screen is lost. This procedure is convenient with small amounts of input-output, for example when working with an interactive program in the "question-answer" mode. Second, the information on the screen may be printed out on the special "VT-343" ATsPU [alphameric printer] or one similar to it by pressing the [PRINT] key (the display is placed OFFLINE), and then the screen is erased by the [ERASE] key in expectation of output. And third, when programming problems that give output, output size limitations can be considered and the output control facilities discussed below can be used.

Expanding the Capabilities of Programs for Input-Output Control

Since certain "Videoton" display control symbols are fully legitimate symbols in the APLVIDEO language, they may be used in text constants and programs, which yields new capabilities for controlling input-output and graphic programming.

Let us discuss several examples. In the examples, we use the following text variables to control the screen to which values have previously been assigned:

ERASE - erase screen and set cursor at home;

HOME - set cursor at beginning of screen;

LF - line feed (blinking symbol J);

ETX - end of text (blinking symbol C);

CD - cursor down one position;

CU - cursor up one position;

CL - cursor left one position; and

CR - cursor right one position.

Example 1. Output text and erase screen

[3= Z = symbol for specification or assignment; see table 3]

```
TXT 3 '*---SAMPLE OF OUTPUT TEXT---*'
ERASE, TXT
```

in the process on a clear screen on the first line will appear the information:

```
*---SAMPLE OF OUTPUT TEXT---*
```

Example 2. Output text at screen line desired without erasing screen

[Russian ∇ = nabla, function definition symbol;
Russian ⍴ = rho, shape/reshape; see table 3]

```
∇ LINE N
[1] HOME          - return cursor to home position;
[2] (N-3)⍴CD      - move cursor down needed number of lines taking automatic line
                    feed into account;
[3] TXT           - output of text;
∇
LINE 8            - call of function which outputs value of variable TXT on line 8
                    of screen.
```

Example 3. Input of text on any screen line and output with storage of line feed

[␣ = blank]

```
POINTER 3'␣␣␣␣␣<---INPUT NAME'      input pointer
LIST 3''                            empty vector of list of names
          ∇ INPUT N
          [1] ERASE
          [2] (N-3) ⍴ CD
          [3] POINTER, CU
          [4] NAME 3 ⎕&'            [  ⎕&' = ⍞  ]
          [5] LIST 3 LIST,NAME,LF
             ∇
```

Comments on statements:
POINTER and LIST are variables defined in the mode of immediate execution of APL;
N is the screen line number, reading from top, on which will be output the phrase POINTER and in the first position of which will be input the name NAME;

1 - 2 -- setting cursor at needed line;
3 -- output of text-pointer of input and compensation for automatic line feed;
4 -- input of text information in field on screen indicated by pointer and entering of it in NAME;
5 -- adding to list of names by next name with storage of line feed.

With repeated operation of the program, a list of names is filled in that can be output in the form of a column of names of operations: LIST

Example 4. Input of information from display screen using SEND mode

Let SCREEN be a text variable containing several lines of information which have to be corrected to suit the job requirements.
The program CORR will make it possible to do this operation:

```
CORR ⍕CREEN
  ∇ CORR TX;SC                          [∇ = nabla, ∇ ]
[1]  SC ← TX                            [← = ←       ]
[2]  ERASE,SC,LF,ETX
[3]  HOME
[4]  12 ⍴ CD                           [⍴ = rho, P   ]
[5]  '⍬⍬⍬PUT DISPLAY OFFLINE, PERFORM CORRECTION AND SEND FROM FIRST LINE',HOME
[6]  SC ← ⎕&'                           [⎕&' = ⍞      ]
[7]  TX ← SC
[8]  ERASE,TX
     ∇
```

Comments on statements:

1 - copy text information into temporary variable SC;
2 - output text, which is text-sample to be corrected, to screen;
3-5 - output message on line 16 and set cursor home;
6 - stop program for input. At this time, display is put OFFLINE, offline correction is made, and then screen is read in SEND mode;
7 - return value to initial variable;
8 - check reproduction of corrected variable on screen.

Example 5. Random walk of cursor over screen starting in its center

```
     ∇ RANDCR N;I;CONTRL;COMPAS         [∇ = ∇ ]
[1]  I ← 0                              [← = ←]
[2]  CONTRL ← ''
[3]  COMPAS ← CL,CR,CU,CD
[4]  CONTRL ← CONTRL,COMPAS[?4],'+'
[5]  I ← I+1
[6]  →I×⍳(I < N)                        [→ = →; × = X (multiplication); ⍳ = iota]
[7]  ERASE;'NUMBER OF MOVEMENTS OF CURSOR=';N
[8]  HOME
[9]  7 ⍴ CD                             [⍴ = rho]
[10] (40 ⍴'-'),CONTRL
     ∇
```

Comments on statements:

1-6 - form control vector CONTRL which includes symbols for cursor movement, and character '+' for indication of its positions on screen;
8-10 - set cursor in screen center and control walk.

Implementation of Working Version of APL System

The APL system in the initial version was implemented on a YeS-1022 computer with YeS OS in the MFT mode. A YeS-7077 console was used as the system terminal; during operation with APL it was not used as the computer operator console.

The display version of the system was implemented with the group switch and 15 "Videoton-340" terminals. The APLVIDEO system operates under YeS OS control in the MVT mode. In both cases, the system uses one priority partition (MFT) or region (MVT) with a size of 180K. The YeS-5050 and YeS5061 devices have been used as external storage. In doing so, for the YeS-5061, 2 extents of sets were used for user libraries with a size of 20 cylinders each and 8 cylinders for the swapping support set.

Practice of operating with the APL system oriented to the YeS-7077 terminals has shown that these devices can be used for APL without alteration. However, the existing restrictions in the modified version of the language could be reduced if use of the keys VSh (backspace), NS (new line) and PS (line feed) caused generation of the appropriate code combinations.

When operating with 15 users, the APLVIDEO system in the majority of cases provides a response time on the order of 3 seconds with a 36K-workarea for each user; in doing so, use of processor time is negligible and the capability exists for parallel batch processing of user jobs for YeS OS.

In addition to those mentioned, other possible terminals for YeS APL are the YeS-8570 subscriber stations connected by communication lines to a data transmission multiplexor.

References

1. Grenander, U. and Faryberger, V., "Kratkiy kurs vychislitel'noy veroyatnosti i statistiki" [Short Course in Computational Probability and Statistics], Moscow, Nauka, 1978.
2. Petrova, Ye. P. and Saplin, M. S., "Ustroystvo obmena informatsiyey operatora s protsessorom" [Operator-Processor Communication Device], Moscow, Statistika, 1976.
3. Iverson, K. E., "A Programming Language", New York, 1962.
4. Katzan, H. J., "APL User's Guide", New York, 1971.
5. APL\360 User's Manual, IBM, H20-0683, 1969.
6. Everett, F. A., "A Formal Definition of APL Statement Syntax", APL-75 Congress, Riza, 1975.
7. Gilman, L. and Rose, A., "APL: An Interactive Approach", Moscow, Mir, 1979.
8. Pratt, T., "Programming Languages", Moscow, Mir, 1979.
9. Ketkov, Yu. L., "Programmirovaniye na BEYSIKE" [Programming in BASIC], Moscow Statistika, 1978, pp 113-116.

8545
CSO: 1863/229

APPLICATIONS

UDC 658.815:681.32.06

SYSTEM FOR MONITORING METAL PRODUCT DELIVERIES TO MAJOR NATIONAL ECONOMIC CONSTRUCTION PROJECTS

Moscow MATERIAL'NO-TEKHNICHESKOYE SNABZHENIYE, SERIYA 4: PRIMENENIYE MATEMATICHESKIKH METODOV, VYCHISLITEL'NOY TEKHNIKI I ORGTEKHNIKI V MATERIAL'NO-TEKHNICHESKOM SNABZHENII in Russian No 8, Aug 80 (manuscript received 14 May 80) pp 8-9

[Article by I. B. Ayngorn, deputy department chief, USSR Gossnab Main Computer Center]

[Text] The task "Monitoring of Metal Product Deliveries for Construction of Enterprises for Ferrous Metallurgy, Enterprises for Production of Fertilizer, and Facilities Constructed on the Basis of Compensation and Integration Agreements" has been developed by the USSR Gossnab Main Computer Center and the NIISU [expansion unknown] and put into industrial operation at the USSR Gossnab Main Computer Center.

Product nomenclature now monitored includes four classes of metal products: rolled ferrous metal products, steel pipes, iron pipes and industrial metal goods. Data on the amounts of resources actually received are sent monthly by union republic gossnab information centers and those of the main territorial administrations of USSR Gossnab from the construction sites to the USSR Gossnab Main Computer Center over communication lines.

The "Minsk-32" computer is used to receive the data from the communication lines and monitor teletype messages, while data processing is done on the YeS-1040 computer. Three data bases have been built with the YeS-1040 computer, maintained by the "Oka" DBMS: "Catalog," "Projects" and "Ministries." The "Catalog" data base holds permanent data for the task, "Projects" holds planned and actual data on metal product deliveries to each site, and "Ministries" holds summary data on plans for delivery and actual deliveries of metal products by target functions, construction ministries and main territorial administrations of the USSR Gossnab.

The OS input generator is used to monitor data and convert the files into a form convenient for loading into the data base.

"Oka" DBMS facilities were used to build programs to load the information into the data base, summarize it by target functions, ministries and territorial administrations and printout the output form. The output document for each target function contains summary data on the delivery of each type of resource for all projects of the target function, including by each ministry, and within a ministry--by each territorial administration and by projects. In the output document are given data

on the plan for deliveries for the year and for the reporting period, and also data on the actual delivery for the reporting period. Planned for the future is expansion of the nomenclature of monitored products, development of new output forms and tying this task to other tasks associated with supply of materials and equipment for capital construction.

Solving this problem achieves elimination of manual monitoring of the fulfillment of assignments, establsihed annually by the USSR Gossnab, to the territorial agencies for delivery of physical resources to the major projects of the national economy.

Address for inquiries: Moscow, Orlikov per., 5, GVTs Gossnaba SSSR [USSR Gossnab Main Computer Center].

8545
CSO: 1863/245

UDC 681.32:[339.6:662.6/.7]

PROCESSING OF ONLINE DATA ON FUEL RESERVES AT MAJOR USSR ENTERPRISES AND POWER STATIONS

Moscow MATERIAL'NO-TEKHNICHESKOYE SNABZHENIYE, SERIYA 4: PRIMENENIYE MATEMATICHESKIKH METODOV, VYCHISLITEL'NOY TEKHNIKI I ORGTEKHNIKI V MATERIAL'NO-TEKHNICHESKOM SNABZHENII in Russian No 4, Apr 80 (manuscript received 29 Dec 79) pp 2-3

[Article by Ye. G. Torchinskiy, deputy department chief, Main Computer Center for USSR State Committee for Material and Technical Supply]

[Text] The task on processing of online data on fuel reserves at major power stations and enterprises in the country has been worked out and placed into industrial operation at the USSR Gossnab Main Computer Center. The task is designed to receive, process on a computer and output to receivers reports on the availability of fuel reserves at more than 600 major enterprises and power stations of 14 ministries.

This problem is solved in two stages: in the first, special forms of normative provisions containing a list of the enterprises and types of fuel used by them are circulated.

The forms of normative provisions are formatted and printed on the YeS-1040 computer at the USSR Gossnab Main Computer Center. Based on the filled-out forms received from the ministries in September of each year, correction and forming of the library of normative provisions and dictionaries of enterprises and types of fuel is performed on the "Minsk-32" computer. Processing of the online information and report formatting is performed in the second, main stage.

Information is acquired from the enterprises and power stations by the information centers of the territorial agencies of the USSR Gossnab. The information received is recorded on the special form for online reporting. The data on the form is punched on perforated tape and transmitted by teletype to the USSR Gossnab Main Computer Center. The information goes directly into the "Minsk-32" computer and is checked; when errors are detected, information on the errors is printed on the subscriber's teletype.

After the information is received and checked, an information file is formatted and processed and the output documents are printed.

The "Dialog" software system, developed by the Information Dispatch Center of the USSR Ministry of the Coal Industry, the "Minsk-1560" communication channel interface apparatus and the "Minsk-32" computer are used to receive, check, output error information to the subscriber, record on magentic tape and form intermediate files. A COBOL software system for the "Minsk-32" computer, developed at the Main Computer Center, is used to form thk main information files and library of normative provisions and dictionaries, to process information and print the output documents. Data for the problem are also used by the Information Center of the USSR Gossnab Main Computer Center to submit information to the Central Staff of the USSR Gossnab by the "Wang-2200" minicomputer. The "Minsk-32" computer magnetic tape is recopied into the YeS computer codes and then recorded on the MD [magnetic disk] of the "Wang-2200" minicomputer. Consolidated data are output to a video terminal and for printing.

The result of solving the problem is a report on fuel reserves consisting of two parts: consolidated data on fuel reserves for the entire list of enterprises and power stations and types of fuel by each ministry; a list of enterprises and power stations with a critical fuel reserve (three days for the Ministry of Power and Electrification, five days for the others).

Implementation of the task has made it possible to intensify monitoring the development of fuel reserves and to improve expeditious management of the process of distributing fuel to the major enterprises and power stations in the country.

Address for inquiries: Moscow, Promyshlennyy proyezd, 3, GVTs Gossnaba SSSR [USSR Gossnab Main Computer Center].

8545
CSO: 1863/245

PUBLICATIONS

ABSTRACTS FROM JOURNAL 'AUTOMATION AND COMPUTER TECHNOLOGY', JULY-AUGUST 1981

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 4, Jul-Aug 81 pp 97, 99

UDC 681.3.06:519.872

SOME RESULTS OF ANALYSIS OF A PROBABILISTIC MODEL OF TELEPROCESSING SYSTEMS

[Abstract of article by S. F. Yashkov]

[Text] A complete analysis of teleprocessing systems described by the system M] G] 1 is given with a discipline of a uniform partition of the processor in which each request present in the system is serviced simultaneously with the others at a variable rate equal to 1/k in the case of k requests at the given time. A new analytic method of studying a system is developed; it is used to derive stationary distributions and features of basic characteristics. A number of practically important characteristics is presented in a form convenient for calculations. New regularities of system behavior are established that substantially expand the capabilities of applying the dervied results for analysis of computer network models. Bibl. of 9 titles, 1 table.

UDC 681.324

STRUCTURE OF A SYSTEM FOR DATA EXCHANGE BETWEEN PROCESSES IN COMPUTER NETWORKS

[Abstract of article by J. Nemeth]

[Text] Computer network services are effected by processes that exchange data with each other. In a hierarchical system, data exchange between processes has been studied less than at the other levels. This effort is devoted to a study of this question. The model is based on the principle of the "meeting place" both to effect switching and to fully establish communications. "Meeting place" location does not depend on terminal location. Communication control system does not depend on the system effecting data exchange. Practical use is illustrated by examples. Bibl. of 12 titles, 8 figs.

UDC 681.324:519.872

MODEL OF A DISTRIBUTED TELEPROCESSING SYSTEM

[Abstract of article by A. A. Amosov]

[Text] Analytic model of the transport level of a teleprocessing system is discussed that considers the topology of the communications network and the system of information transmission matrices. The communications network is represented as a system of interacting transmission paths (virtual connections) modeled by queueing networks of a special structure. A precise analytic solution is given to the problem of finding the final vectors of states of the paths using the principle of semilocal balance. Bibl. of 6 titles.

UDC 681.32:519.713

CALCULATION OF LOWER-BOUND ESTIMATE OF NUMBER OF TERMS IN DNF SYSTEM DESCRIBING LOGIC STRUCTURE OF AUTOMATON

[Abstract of article by E. E. Lange]

[Text] A simple procedure is proposed for calculating the lower-bound estimate of the number of different terms in a system of DNF [disjunctive normal form] of Boolean functions that describes the logic structure of a finite automaton. The calculated estimate is the best in the sense of achievability. The calculation procedure consists in constructing some finite undirected graph and determining its chromatic number. Reducing graph dimensions in constructing it is considered to simplify determination of the chromatic number. The procedure can be used to estimate the possibilities of implementation of an automaton to be synthesized with a programmable logic array with specfied dimensions and to estimate the quality of solutions to the problem of optimization of the logic structure of an automaton that are derived by using approximation methods of logic synthesis. Bibl. of 7 tiltes, 3 tables.

UDC 681.323:518.5.001

IMPLEMENTATION OF PARALLEL POLYNOMIAL APPROXIMATION IN MULTIPROCESSOR STRUCTURE

[Abstract of article by O. N. P'yavchenko, Ya. Ye. Romm and I. F. Surzhenko]

[Text] A method is proposed to implement in a multiprocessor structure a parallel polynomial approximation in a time proportional to the logarithm of the degree of the polynomial, with a proportionality factor equal to the time of one multiplication. Applications to implementation of functions, division and linear differential equations are given. Variants of apparatus implementation are given. Bibl. of 14 titles, 2 figs., 1 table.

UDC 681.3-181.48

MICROPROCESSOR IMPLEMENTATION OF DIGITAL CONTROL AUTOMATA

[Abstract of article by B. M. Kagan, G. Kh. Novik, V. G. Persheyev and M. I. Shamrov]

[Text] Discussed are features of implementation oriented to logic control of digital control automata with microprocessor systems (M-systems). Different approaches to construction of M-system programs are given. Methods of program implementation of logic converters are analyzed. Bibl. of 5 titles, 5 figs.

UDC 519.873:681.324

APPLICATION OF RELIABILITY THEORY TO PROBLEM OF EXPERT EVALUATION OF COMPUTER SYSTEMS

[Abstract of article by V. I. Levin]

[Text] Discussed is the problem of aggregating individual orderings Y_i of one and and the same set of objects by different experts i = 1, ..., n. In contrast to the classic statement, the author considers the link between Y_i and the "true" ordering of objects Y (by introducing the probability q of error by an expert in evaluating the relationship $\square_{ij}$ between two objects). The author suggests aggregating the evaluations $\hat{\square}_{ij}$ according to the α-majority rule (when α = 0.5 -- according to the majority rule). It is shown that for $\alpha > q$ all these rules when $n \to \infty$ are asymptotically devoid of errors, but only the majority rule with the probability of 1 precludes contradiction or uncertain situations. Aggregating the evaluations Y_i into the evaluation Y is done by summing the collective evaluations of the paired relationships for all pairs of objects. It is shown that for that rule of aggregation, the probability of a collective error of the first kind (transposition of objects in Y) and of the second kind (formation in Y of loops) approaches zero when $n \to \infty$. Bibl. of 4 titles, 1 table.

UDC 681.326.74.06:621.3.049.774.2

METHOD OF CONSTRUCTING CHECKING EXPERIMENTS FOR LARGE-SCALE INTEGRATED CIRCUITS OF MAIN STORAGE

[Abstract of article by V. G. Totsenko and B. V. Strelyayev]

[Text] A method is suggested for constructing checking experiment (PT) for MOS LSI memory based on a general approach called "checking the hierarchy of properties." The checking experiment is specified by two sections of input-output sequences (VVP), one of which is designed to check the electronic framework (EO), and the other to check the memory matrix and read/write amplifiers. When building the first section of the input-output sequences, assumptions are made on the possible nature of manifestation of any malfunctions of the electronic framework. The second section of the input-output sequences is not built in advance: in building the first section, consideration is given to the necessity of including in it sections of the

checking experiment that test the storage elements of the matrix. Completeness of the proposed test is shown. Length of the suggested test for the memory equals $6N + 2N (\sqrt{N} + 1)$ where N is memory size. Bibl. of 9 titles, 1 fig.

UDC 681.324

ENUMERATION OF ELEMENTARY SECTIONS IN A COMPUTER COMMUNICATION NETWORK

[Abstract of article by Yu. K. Apraksin]

[Text] Discussed is the problem of finding the set of all elementary sections in a computer communication network. Used as a model of the computer communication network structure is a graph of communications; its structure is specified by a matrix of incidences. The suggested algorithm for finding the elementary sections between two apexes of the graph is based on using a graph contraction operation and is a simple procedure for transforming the matrix of incidences into the matrix of elementary sections. Bibl. of 7 titles, 3 figs.

UDC 621.376

SIGNAL PROCESSING IN DISCRIMINATOR SYSTEMS CONSIDERING NOISE

[Abstract of article by G. E. Vasarin'sh and E. Kh. Khermanis]

[Text] Discussed are probability aspects of signal processing in discriminator systems with regard to intrinsic noise. Assumed as known are the laws of distribution of additive noise of each discriminator. Probable characteristics of application of discriminator systems are dervied. Examples are cited. Bibl. of 4 titles, 5 figs.

UDC 621.391.244:517.587

SIGNAL ANALYSIS IN TERMS OF REAL EXPONENTS

[Abstract of article by A. A. Kochetkov, V. V. Krylov and D. M. Ponomarev]

[Text] Signal analysis by exponents with real indexes is discussed. Described is a method of determining the signal relaxation spectrum based on constructing an inverse filter in aggregation with a Fourier transformation. The method permits relatively simple construction of a digital analyzer of the relaxation spectrum. An algorithm is given to calculate spectral density of a signal specified by its discrete samples using a fast Fourier transform. Bibl. of 4 titles, 3 figs.

UDC 681.324

RATE CAPABILITIES OF SYNCHRONOUS INFORMATION TRANSMISSION BY 'CHANNEL-TO-CHANNEL' ADAPTERS

[Abstract of article by V. A. Red'ko and F. A. Sklyarevich]

[Text] Discussed is synchronous process of indormation transmission using a "channel-to-channel" adapter having a sectionalized buffer with limited storage.

The mathematical model of the system in question is an SMO [queueing system] in which request service time and time between sequential entries of requests have constant components. A technique is offered for approximate analysis of such queueing systems to derive estimates of rate capabilities of synchronous information transmission. Example of application of technique is given. Bibl. of 5 titles, 4 figs.

UDC 681.324

SYNTHESIS OF MULTIMACHINE STRUCTURES FOR AUTOMATION OF SCIENTIFIC RESEARCH

[Abstract of article by G. P. Vasil'yev, G. A. Yegorov and V. I. Shyaudkulis]

[Text] Based on an example of a specific multimachine hierarchical complex, authors discuss solution to problems of synthesis of logic and physical structures of such systems. They substantiate composition and structure of system software and organization of interaction of its various elements. They discuss interfaces between user programs and system software. They point out the possibility of using the suggested logic structure in complexes of various configurations. Bibl. of 8 titles, 7 figs., 1 table

8545
CSO: 1863/248

COLLECTION OF PROBLEMS ON MACHINE PROCESSING OF ECONOMIC INFORMATION

Moscow SBORNIK ZADACH PO MASHINNOY OBRABOTKE EKONOMICHESKOY INFORMATSII in Russian (signed to press 23 Feb 81) pp 2, 109, 132-133, 135

[Annotation, table of contents, tables 18 and 36 from book "Collection of Problems on Machine Processing of Economic Information", by Gerta Kirovna Begotskaya, Vasiliy Petrovich Kosarev, Boris Mikhaylovich Rudzitskiy and Emma Alekseyevna Umnova, Izdatel'stvo "Finansy i statistika", 10,000 copies, 136 pages]

[Excerpts] This book contains the problems associated with the design of systems for machine processing of economic information--SMOEI: design of codes, source documents and machine media. The main emphasis is on development by design of computer installations (estimate design work). The varieties of information processing systems are treated and problems on evaluating the efficiency of different computer data processing systems are elaborated.

This book is intended for students in VUZ's specializing in economics.

Table of Contents

Table 18. Table of Computer Center Categories by Staff Wages

Quantitative and Qualitative Characteristics		Number of Points 2	4	6	8	10	12
Volume of work performed, thousands of rubles	1. to 200		X				
	2. 200-500			X			
	3. 501-800				X		
	4. 801-1200					X	
	5. over 1200						X
Problem composition by performance schedule	1. unlimited	X					
	2. current		X				
	3. on-line			X			
Functions performed	1. problem programming and algorithmization		X				
	2. automation of programming and introduction of economic-mathematical methods			X			
	3. organization of ASOD [automated data processing system], methodological direction and coordination of work of related computer centers					X	

Classification of computer center to wage groups	to group I	over 18 points
	to group II	from 11 to 18 points
	to group III	to 10 points

Table 36. Standards for Estimating Labor Input and Cost of Operations

Type of Operation	Type of Equipment	Unit of Measurement	Average Hourly Output Norm (N)	Operator's Hourly Rate, Rubles (Ch)	Hourly Norm of Amortization, Rubles, (Ch_a)
1	2	3	4	5	6
1. Document reception		document line	700	0.4	--
2. Pricing	Iskra-111	operation	650	0.34	0.02
3. Pricing supervision	"	"	750	0.34	0.02
4. Check pricing by supervisory tabular form "Correction of Errors"	"	"	6,000	0.4	0.02
5. Automatic pricing	EVP80-2	card	5,000	0.4	0.215
6. Automatic card selecting or merging	RPM80-2M	"	15,000	0.34	0.294
7. Card punching	P80-6	"	220	0.4	0.016
8. Alphameric card punching	PA80-6	"	200	0.4	0.055
9. Check card punching by verification	K80-2	"	290	0.4	0.016
10. Check alphameric card punching by verification	KA80-2	"	250	0.4	0.032
11. Addition on tape	SDV-107	operation	1,200	0.34	0.004
12. Counting method of verifying punched cards--comparison of totals, correction of errors	SDV-107	row-column (word)	800	0.4	0.004
13. Counting method of card punching verification--combination of operations 11 and 12	SDV-107	"	800	0.4	0.004
14. Reproduction and duplication of cards	PR80-2	card	5,000	0.4	0.2
15. Interpretation of cards	RM80-1	"	4,500	0.4	0.3
16. Perforation by graphic marks	PS80-1	"	5,000	0.4	0.137
17. Repunching of erroneous cards	P80-6	"	100	0.4	0.016
18. Balancing tabulation "for printing"	T-5M	"	4,200	0.4	0.15

19. Balancing tabulation with pricing	T-5M	"	4,000	0.4	0.5
20. Balancing tabulation "for total"	T-5M	"	6,000	0.4	0.15
21. Checking of balancing tabular forms and entry of check sums in journal	Iskra-111	batch of documents	50	0.4	0.02
22. Card sorting	S80-5M	card column	20,000	0.34	0.19
23. Card sorting	SE80-3	"	28,000	0.34	0.055
24. Tabulation of summaries "for printing"	T-5M	card	3,100	0.4	0.15
25. Same, with total punching	PI80-1	"	2,800	0.4	0.207
26. Tabulation of summaries "for total"	T-5M	"	3,500	0.4	0.15
27. Same, with total punching	PI80-1	"	3,200	0.4	0.207
28. Tabulation of alphameric summaries	TA80-1	"	4,200	0.4	0.35
29. Same, with total punching	PI80-1, TA80-1,	"	3,900	0.4	0.407
30. Output of tabular form summaries	Iskra-111	tabular form	1	0.4	0.007
31. Tape punching	STA-2M	word	2,800	0.4	0.025
32. Automatic collation of two tapes	KSU-1	"	15,000	0.4	0.048
33. Automatic copying from tapes to cards	BLP-1	card	4,800	0.4	1.125
34. Card input to computer	YeS computer	"	30,000	--	--
35. Card output from computer	YeS computer	"	6,000	--	--

Note. In operations 12, 21 and 30, the models of the machines are used in a limited way. But in estimating their amortization, it is necessary to account for full labor input of the operations since at this time they are not used for other operations.

For all machines, except the reproducer, collating and interpreting machines, electronic computer perforator, reading perforator and devices for compying from perforated tapes to cards, the hourly amortization norm is based on two-shift operation.

8545
CSO: 1863/257

UDC 681.32:621.32

MULTIFUNCTIONAL AUTOMATA AND THE ELEMENT BASE OF DIGITAL COMPUTERS

Moscow MNOGOFUNKTSIONAL'NYYE AVTOMATY I ELEMENTNAYA BAZA TSIFROVYKH EVM in Russian 1981 (signed to press 18 Dec 80) pp 2-4, 239-240

[Annotation, foreword and table of contents from book "Multifunctional Automata and the Element Base of Digital Computers", by Valentin Aleksandrovich Mishchenko, Valeriy Dmitriyevich Kozyuminskiy and Aleksandr Nikolayevich Semashko, Izdatel'stvo "Radio i svyaz'", 6,000 copies, 240 pages]

[Text] Presented systematically in this monograph are the theoretical and practical aspects of the synthesis of general-purpose and multifunctional digital automata--a new class of digital devices that permit implementation of specific functions based on the principle of multifunctionality. The element base, implemented on the basis of these automata, permits building high-speed computers and homogeneous computing media. Described in the book are methods for synthesis of these automata, and numerous examples of implementation of concrete structures are given.

The monograph is intended for scientific workers, engineers, post-graduate students and those in upper-level courses of VUZ's studying the questions of computer technology and electronics.

Foreword

The great and critical problems facing domestic computer engineering are forcing specialists in this field to look for new methods of synthesis of computer structures. It is now clear that the future belongs to multiprocessor computer systems for parallel processing of information, in which the questions of regularity of the medium play an essential role, affecting the durability, reliability, adaptability to manufacture and a number of other practically important indicators of computer structures.

The abundant possibilities of contemporary and future microelectronics are setting an economically important task for theoreticians and practitioners: development of a set of LSI circuits that will meet the needs of a broad range of users and take into account the future development of computer structures. In extensive use now as these LSI circuits are microprocessors in combination with programmable logic arrays.

In this book, the authors offer the reader another class of automata that can be used both in combination with programmable logic arrays and microprocessors and as stand-alone units. These are general-purpose and multifunctional automata with practically unlimited functional capabilities that permit building super fast computers and homogeneous computer media. Preliminary studies made by the authors of this book have indicated their practical implementability [57-60].

The very problems of general-purpose and multifunctional structures cannot be called absolutely new. In the domestic and foreign literature, there are a number of largely theoretical works on this topic. These are the works of Soviet experts: E. V. Yevreinov, I. V. Prangishvili, E. A. Yakubaytis et al., as well as of foreign authors [53-56]. Nevertheless, the authors believe that this relatively new direction is in need of certain popularization and deserves greater attention from developers.

In this book, the authors present their own original conception of general-purpose and multifunctional structures and cite fundamentally new provisions and results of synthesis of general-purpose and multifunctional automata. In contrast to known works, the authors have set themselves the task of bringing theoretical results to practical schemes and techniques for their synthesis that can be used in design work. At the same time, the book has a clearly expressed monographic nature since it generalizes a number of known works by the authors listed above and presents systematically the main achievements in this field.

The features of general-purpose and multifunctional automata that afford a strict mathematical description of the process of their synthesis permit solving a number of known problems that now bother computer designers and developers. Here are some of them:

1. The large nomenclature of LSI circuits required for building modern computers. General-purpose and multifunctional automata permit reducing this nomenclature to the minimum for many ideologies for building architectures for future digital computers. Solving this problem has a clearly expressed economic nature since development of modern LSI circuits is rather expensive.

2. Increasing the output of suitable circuits in series production. This problem is solved by the broad functional capabilities of the element base, which permits improving the technology of production that does not vary over an extended period and thereby increasing the output of suitable circuits.

3. Sharply reducing the volume of reserve components and raising the repairability, durability and reliability of computers. This problem is solved by developing interchangeable modules of minimal nomenclature with controllable functionality with invariable structure and links, which ultimately permits putting reserve elements in the machine itself.

4. Machine synthesis of elements and devices for computers. This problem may be solved on the basis of the theory of general-purpose and multifunctional automata through the properties of universality of these automata, since a general-purpose module may "generate" a large number of digital microstructures.

The reader will find in this book a more detailed presentation of the individual parts of these problems.

This book was written by a team of authors. The foreword and sections 1.1, 2.4, 2.5, 5.7 and 10.3 were written by V. A. Mishchenko, V. D. Kozyuminskiy and A. N. Semashko; chapters 2, 3 and 7-10 were written by V. A. Mishchenko and V. D. Kozyuminskiy; and chapters 1 and 4-6 were written by V. A. Mishchenko and A. N. Semashko.

The authors and publisher will be very grateful for comments and remarks on the book's content; these should be sent to: Moskva, Glavpochtamt, a/ya 693 [Moscow, Main Post Office, Box 693], "Sovetskoye radio".

Table of Contents

8545
CSO: 1863/257-A

UDC 681.32

PARALLEL PROCESSORS FOR CONTROL SYSTEMS

Moscow PARALLEL'NYYE PROTSESSORY DLYA UPRAVLYAYUSHCHIKH SISTEM in Russian 1981 (signed to press 3 Apr 81) pp 2-11, 142-158

[Annotation, table of contents, preface, introduction, chapter 9 and bibliography from book "Parallel Processors for Control Systems", by Yakov Il'ich Fet, Energoizdat, 12,000 copies, 160 pages]

[Text] Author deals with questions associated with algorithmic and logic structures of parallel processors. He covers various types of homogeneous processors that make it possible to substantially raise the speed of systems in solving large-scale computational and informational-logic problems.

The processors described can be used for hardware implementation of software functions and for developing various functional-oriented systems.

For engineers and technicians engaged in design of computers and systems, as well as students in VUZ's of the corresponding specialties.

Table of Contents

Preface

In modern control systems, the amount of information and requirements for fast processing of it are continually increasing.

One important reserve for increasing the efficiency of control systems is the use of special-purpose digital devices and, in particular, parallel processors.

The achievements of modern microelectronics are turning parallel computations and parallel processors into reality. A number of efficient associative and conveyor type parallel processors have emerged. Use of them shows that the effective speed in solving suitable problems is increased several orders.

The heightened interest in special-purpose and parallel processors has also been reflected in the increasing number of articles published in Soviet and foreign scientific and technical journals.

In this book, special attention is paid to one type of parallel computing device--the so-called special-purpose homogeneous processors, covered in the first four chapters. Each special-purpose homogeneous processor is a two-dimensional combinational iterative array, matched with matrix storage. The processor is oriented to execution of some group (large-block) operation on an array stored in its storage elements. These basic operations can have independent value (for example, in information retrieval problems) or serve as the basis for microprogram implementation of various algorithms.

Also discussed are other types of parallel processors, in some sense "related" special-purpose homogeneous processors, namely: associative (chapter 5), orthogonal (chapter 6) and dynamic (chapter 7).

In connection with organization of parallel processing of large arrays of information in modern control systems, assuming great importance are questions of rearrangement of arrays and conversion of their structure. The possibility of using some special-purpose homogeneous processors for data structure conversion is shown in chapter 8.

The description of the specialized processors is illustrated by examples of solving individual large-scale problems that are typical for control systems. Questions of application of parallel processors in control systems are also discussed in chapter 9.

The author is sincerely grateful to L. V. Kantorovich and D. A. Pospelov for the great interest and attention shown to this work.

The author thanks V. N. Aleyeva, N. A. Zosimova and Ye. V. Suvorov and the editor, Ye. I. Piyl', for their valuable comments that facilitated improving the content of the book.

The Author

Introduction. Control Systems and Problems of High-Volume Information Processing

The modern scientific and technical revolution is inconceivable without the widespread application of various automated control systems. The necessity of continual development of control systems, and increasing their functional capabilities, precision and reliability of operation makes enhanced demands on digital computing devices--the major component of these systems.

The requirement for enhanced efficiency of computers is closely tied to the features of the problems solved in control systems [1].

Typical for automated systems of organizational control is the extremely large volume of information (data bases used in large control systems are continually growing and may reach 10^{10} to 10^{11} bytes). Modern computers (third-generation machines), as a rule, cannot adequately handle the processing of such large files.

In automated and automatic process control systems, the volume of information to be processed is relatively not large, but usually processing of control actions is required in the real-time mode. With the large rate of flow of processes in the controlled object, this exceeds the capability of existing machines.

Finally, special-purpose automated systems can make very high demands on the volume and speed of processing. An example of a special problem with such demands is air traffic control at a modern airport (recall that the STARAN special-purpose processor, the most powerful of digital machines now in operation, was developed just to solve such problems [2]).

Thus, there is now a discrepancy between the requirements of the problems to be solved in control systems and the capabilities offered by computers in series production.

Practical ways of raising computer efficiency are well known: these are raising the speed of the elements, improving software and developing the logic structure and architecture of the machines.

As for element speed, almost all possibilities have been exhausted (discounting optical elements that may emerge in future generation systems). Experts believe that we can expect only a one-two order of magnitude further increase in computer efficiency by increasing the speed of the electronic elements.

Software improvement is aimed basically at speeding up preparation of the problems (automation of programming, application of problem-oriented languages), but not the solution of them. Actually, on the one hand, the modern operating system facilitates fuller use of computing system resources, and as a consequence of this, an increase in its efficiency. On the other hand, complex software systems require for their "own" purposes considerable consumption of storage and time, which reduces the overall positive effect. Also, development of these systems entails large inputs of human labor which leads to high cost of them (especially compared to the rapidly falling cost of hardware).

The most important reserve for raising the efficiency of computers is, apparently, the evolution of their architecture. This stems from the majority of problems to be solved in control systems having a special nature; the initial information of these problems is organized into special structures (arrays, tables, etc.). In solving these problems on general-purpose machines having classical architecture that are oriented to a standard set of basic instructions and operate sequentially with scalar values, it is not possible to take advantage of the specific nature of the data and types of processing.

Proposals on developing special-purpose software and hardware to solve complex problems with a special nature have been made relatively long ago [3, 4].

The question of expanding the internal machine language by raising the hardware level of interpretation and matching computer instructions with high-level language statements was considered in detail in [5].

Applying the terminology introduced in [6], one may say that machines with classical architecture have an "elementary education"--within the limits of arithmetic in the early grades. Machines will operate considerably more efficiently if their "education" level is raised--if they are taught hardware execution of the more complex and special functions from among those now executed by software (for example, calculation by formulas, implementation of cycles, retrieval of information, editing of data, execution of some functions of operating systems).

Improving machine architecture and including special-purpose devices oriented to execution of various complex functions (in particular, parallel processes) in the makeup of the machines entails substantial hardware outlays. Until recently, this was an obstacle to practical use of this important reserve for increasing efficiency. However, the rapid development of integrated circuit technology is creating practical prerequisites for the design and manufacture of machines with a new architecture.

It is well known that the degree of LSI doubles every one to two years while circuit cost goes down annually by about 30 percent. This makes it possible to produce very complex hardware with very small dimensions at low cost. Thus, in 1978, Zilog Inc. began marketing a 16-bit single-chip microprocessor, the Z8000, with about 17,500 transistors on a chip the size of 6 x 6.5 mm [7]. Other firms produce similar microprocessors also. By the mid eighties, forecasters expect development of a 32-bit computer with a degree of integration of 100,000 transistors per chip [8]. Even greater densities are being achieved in storage devices. A number of firms are already producing random-access memory of 64K bits on a single chip, and by the mid eighties, 1M-bit units are expected [9].

Microprocessor LSI circuits will apparently become the main element base for computers in the next few years.

It is necessary to note that in an ideological, structural sense, all microprocessor sets (microcomputers) now being produced remain machines with classical architecture--serial operation. The majority of them are relatively simple devices, approaching in the best case in characteristics third-generation minicomputers.

In the process, the very conservative structural solutions used in the machines made with the new elements do not permit taking advantage of the great possibilities contained in the new technology.

Evidently, favorable conditions have arisen for a serious reexamination of computer architecture. The possibilities of raising the "education" level of the machines by raising the hardware level, specialization of individual devices and application of high-volume, group and parallel processing have become concrete.

Compared to the other ways of improving computers, development of architecture is especially important since it affords new possibilities for construction of algorithms and stimulates development of better mathematical approaches to solving

problems. The emergence of the first parallel processors has led to a considerable number of efforts in which computational methods and algorithms are being revised to fit the capabilities of the new machines [10]. The importance and promise of such efforts are noted in [10]. Experts believe that coordinated, interrelated development of computational methods and structures of computers can lead to a three-four order of magnitude increase in efficiency of solving many problems, including the major problems to be solved in control systems.

What place in the development of computer architecture do the parallel processors discussed in this book hold? To answer this question, we have to make some sort of classification of digital systems. M. Flynn [11] singles out four types of systems in his classification that has received wide dissemination:

1. With one stream of instructions and one stream of data (OKOD); classical single-processor machines (von Neumann architecture).

2. With one stream of instructions and many streams of data (OKMD). The same sequence of instructions is executed simulataneously on many sets of arguments. These systems are best adapted for execution of component-by-component vector operations. This type includes the majority of the so-called "matrix" and "vector" processors (array processors).

3. With many streams of instructions and one stream of data (MKOD). The single stream of data is moved along a "conveyor," at each position of which is executed one instruction (operation) from the specified sequence. The stream of results is removed from the output of the "conveyor." MKOD systems have been termed mainline or conveyor systems.

4. With many streams of instructions and many streams of data (MKMD). In this system, several programs are executed simulataneously with independent data (although exchange of information between inidvidual elements of the system is also possible). The MKMD type includes multiprogram multiprocessor systems, computer networks and some array processors.

Of these four types of systems, the OKOD are sequential types of machines; the others employ various forms of parallelism.

In accordance with this classification, the parallel processors given further consideration here are of the OKMD type. Machines of this type, however, can be built differently. Thus, for example, a homogeneous computer system (network) consisting of a number of general-purpose computers executing the same program may also fall within the OKMD type. We are interested in a particular type of parallel processor--processors with a homogeneous internal structure (microstructure). Each of these processors is based on some special-purpose homogeneous processor--a two-dimensional homogeneous structure oriented to implementation of a particular special function and an information processing algorithm. From the viewpoint of the microstructure, a special-purpose homogeneous processor is a variety of a homogeneous computer environment [12]. As a rule, it is a two-dimensional combination iterative array [13] matched with two-dimensional storage.

In a functional sense, a special-purpose homogeneous processor is a problem-oriented digital device (automaton). After the source information (arguments) is input into the processor storage elements and the boundary constants are fed to its external inputs, results are generated after some time (depending on the duration of the transient processes) at specific external outputs.

Let us call the operation that the special-purpose homogeneous processor executes in similar fashion (in one cycle) a basic operation, and the set of control signals that implement this basic operation a micro instruction. It will subsequently be shown that the same special-purpose processor may execute several basic operations (micro instructions). From these micro instructions, micro programs are compiled to solve various problems more complex than those embedded in the processor structure as its basic operations.

Since each basic operation is an operation on an entire array of data, the micro programs of special-purpose homogeneous processors in their speed are considerably more efficient than standard subroutines by which data is similarly processed in OKOD type machines.

Thus, a special-purpose homogeneous processor consists of an operating unit (the two-dimensional homogeneous structure proper) and a microprogram control unit. When operated within a control system, it can be considered a special-purpose peripheral. In the process of executing some complex program, the central machine in the system (the general-purpose computer) sends to the special-purpose homogeneous processor specific group statements and corresponding arrays-arguments. After receiving the task, the special-purpose processor executes it autonomously and outputs the results.

Given in [14] is a detailed classification of parallel processors, according to which devices similar to special-purpose homogeneous processors (in particular, the so-called Kautz arrays) are classed as associative processors with a high degree of parallelism.

It is shown in chapter 4 that there is another link between special-purpose homogeneous and associative processors (associative storage units), and with that it is suggested that an associative processor be considered a special case of a special-purpose homogeneous type.

In addition to associative, we shall also discuss other variants of parallel processors--vertical, orthogonal and dynamic, and the links between them and the corresponding special-purpose homogeneous processors will be shown.

Chapter 9. Possible Applications of Parallel Processors in Control Systems

31. Autonomous Special-Purpose Processors

From the preceding chapters, it is evident that special-purpose homogeneous processors and the other parallel processors can efficiently solve the various problems that are typical for control systems.

Special-purpose processors are now used as autonomous special-purpose computing and control units, as special-purpose attachments operating in a complex with a standard (general-purpose) computer and as functional modules within multiprocessor data processing systems.

Let us discuss in more detail some features of these applications of special-purpose processors.

As autonomous units, special-purpose processes are widely applied in on-board and ground special-purpose systems, in control machines for some processes and in systems that control operations. Usually of decisive importance in this case are the small dimensions and weight of the processor as well as the speed of decision-making.

In these cases, special-purpose homogeneous processors must be used in the mode of executing their own basic operations.

An example is the application of the processor with the basic operation PNB [maximum search] for recognition of critical situations in controlled objects. Let us assume that the values of some parameter are continuously coming into the storage elements of an $_{max}$-processor from a number of controlled objects. Signals at the right boundary of the processor will always indicate that object at which the regulated parameter reaches maximum value.

Instead of an a_{max}-processor, a ∇_{min}-processor, adjusted to execute the maximum search operation, can also be used for this purpose. Further, if the maximum permitted value of the regulated parameter is sent to the register of the boundary attributes of the ∇_{min}-processor, then using the basic operation PVB [search for all greater] (or PVM [search for all lower]), the objects whose parameters are outside the tolerance can be dynamically identified.

A similar problem--dynamic identification of an object with the maximal parameter is solved in the unit for allocation of priority in a multichannel data processing system.

In the implementation of walking robots [71], the coordinates of the robot's legs must continuously be compared to the coordinates of obstacles in the process of constructing the path of its movement to avoid collision with the obstacles. This problem can be solved using a ∇ -processor the following way. For each leg, let the differences between its current coordinate and the corresponding coordinates of all obstacles be calculated continuously (let us note that this calculation is a component-by-component operation between a scalar and a vector). The differences obtained go into the rows of the ∇ -processor, and the minimal permitted difference is recorded in its register of boundary attributes. In doing so, it is evident that a PVM [search for all lower] operation will dynamically select the row corresponding to a dangerous situation--impermissible approaching of a leg to an obstacle, and identify this obstacle. High-volume, parallel processing makes its possible to control the path of robot movement with a large number of legs and obstacles in real time.

Another example of an autonomous special-purpose homogeneous processor: ρ -processor that solves the problem of sorting (grading) items of mass production. If the boundary values of the permitted zones are recorded in the rows of a ρ -processor and the values of the measured parameter of the items are sent to the buses a, the signals generated at the right boundary can be used to control the receiving mechanism of an automatic sorter.

32. Special-Purpose Attachments

These attachments substantially increase efficiency if the special problems to which they are oriented make up a considerable portion of the load on the computer complex. Many examples of successful use of special-purpose attachments both for solving complex computational problems and for information-logic processing of large arrays are well known.

The IBM 2938 special-purpose processor is used in a complex with the 360 (370) machines. This processor is oriented to executing a number of high-volume operations: component-by-component addition and multiplication of vectors, scalar product of vectors, reductive addition and multiplication and others. Based on these operations, efficient micro programs are built to solve differential equations, problems of linear algebra, correlation and spectral analysis, etc.

The category of special-purpose attachments of a computing orientation should include the vertical processor of the OMEN [Orthogonal Mini EmbedmeNt] system (attachment for the PDP-11 machine) and the array processor (quadrant) of the ILLIAC-IV system (attachment for the B-6500 machine).

Several examples of the use of special-purpose processors oriented to information logic problems were given in chapter 7.

The application of a general-purpose computer in a complex with a special-purpose sorting processor for ordering of large files was discussed in detail in [72, 73].

The special-purpose sorting processor consists of the following units: control unit, storage unit for attributes (associative type), storage unit for addresses and the micro program control unit. The processor is considered an active peripheral connected to the selector channel of a general-purpose computer. Access to it is organized as to an external device with a suitable number. The controlling program in the general-purpose computer prepares in main storage the file to be processed as well as the following information needed for operation of the special-purpose processor: A_1 -- beginning address of file M in main storage; a_1 -- beginning address of attribute of first record in file M; l-record length; r- attribute length; n - number of records in file M; f - special-purpose processor operation code.

When the special-purpose processor is accessed, this information is recorded in the control unit. Then independent operation of the processor begins and it consists of two stages: loading and processing proper. In the loading stage, the control unit sequentially for each M file record calculates its beginning address A_i and the beginning address of its attribute a_i. The attribute is called from computer main storage and entered into the next row of attribute storage, and the address A_i into the corresponding row of address storage. After attribute storage (and address storage) is filled, the processing stage begins. At each step of this stage, the specific row corresponding to the given operation code f is selected in attribute storage (for example, the row with the minimal attribute if ascending order is executed). The beginning address of the record with the selected attribute is sent from address storage to the next cell of computer main storage. Thus, in the

process of the processing, there is formed in main storage a table of beginning addresses of file M records ordered by the values of their attributes. Based on this table, the controlling program in the general-purpose computer copies the records of file M in the needed order.

As indicated in [74], by using the special-purpose attachment, the efficiency of the system in solving various information retrieval problems is increased by one to two orders of magnitude. Similar assessments of efficiency were obtained as a result of modeling of a data base processor [57].

The simplest associative memory was used as attribute storage in the attachments described in [72, 73]. If a special-purpose homogeneous processor with a higher hardware level and greater functional capabilities (the ∇ -processor, for example) is used, then obviously greater efficiency can be obtained and the range of problems solved expanded.

33. Functional Modules

The decrease in cost of LSI circuits and the increase in level of integration makes it possible to raise the question of more extensive use of special-purpose processors, paticularly of including them as functional-oriented modules with various data processing systems.

It is believed that in this case each functional module performs some procedure assigned to it, thereby implementing some subroutine by hardware. If an anticipatory review of the overall program is made and data directed for processing to the appropriate functional modules, then the central processor of the system is released from executing a number of subroutines and, what is especially important, from inefficient sequential execution of cycles in implementing high-volume operations. In this system, hardware supports the software.

A natural question arise: how many different functional modules is it necessary to have in a system?

It is well known that a special-purpose processor executes most efficiently that operation to which it is oriented by hardware. There is a contradiction between the attempt to perform each high-volume operation in a functional module designed especially for it and the necessity of a reasonable limitation on the nomenclature of modules. This contradiction can probably be resolved if the following circumstances are considered.

First, each special-purpose homogeneous processor, as a rule, implements not one, but several basic operations; second, in this case, the specific content of a particular operation does not play a role, but only the nature of the computing process necessary for implementation of it. By type of computing process (and consequently, also by requirements for hardware of a functional module), several different operations may coincide. Thus, in a certain sense, all component-by-component arithmetic operations on vectors are similar. Several elementary functions (for example, sin x, tg x and others) can be implemented in one device, that, for example, is oriented to computation of polynomials with different coefficients.

To estimate the number of functional modules that afford satisfactory system efficiency, high-volume statements or procedures can be classified in some way. To this end, the composite and mixed functions of the APL language was taken as a list of high-volume statements [46, 75]. This selection is justified because, for example, all high-volume statements in the PL/1 language (built-in functions, expressions with arrays) are included in the list of APL statements. The same may be said of many procedures, used practically in programs, but not having corresponding statements in high-level languages.

The analysis of high-volume statements in APL in [75] showed that by nature of processing, they can be grouped into the following five classes.

1. Component-by Component Computational.

This class includes mainly high-volume statements in which the arguments are one or two (or more in rare cases) numeric vectors, the result is also a numeric vector and the processing consists of executing a more or less complex transformation of a computational nature. This transformation is applied independently to each component (or set of similar components) of vectors-arguments and may in principle be performed aimultaneously on all sets.

Best suited for execution of statements of the first class are special-purpose processors of the associative type (for example, the π -processor).

2. Reductive Computational.

This class includes high-volume statements in which the argument is a numeric vector, the result is a scalar and the processing consists of sequential application of some computational operation to the components of the vector-argument. Examples of these operations are reductive addition of an array and reductive multiplication. High efficiency in executing reductive computational operations is achieved in conveyor (mainline) parallel processors [76].

3. Component-by-Component Logical.

This class includes mainly high-volume statements in which the arguments are two numeric vectors, the result is a logical vector and the processing consists in executing some logic operation (for example, comparison) on some pair of similar components of vectors-arguments. Processing may be performed simultaneously on all pairs of components.

The η -processor is best suited of the processors described above for execution of component-by-component logic operations.

4. Reductive Logical.

This class includes high-volume statements in which the argument is an array of binary numbers (or bit strings); the result is a subset of the elements of the array-argument (in the particular case, a number or row) associated with the initial array by a specific relationship corresponding to the given statement. Processing consists in executing on an argument some global operation that can be called a "scan" of the array.

Examples are the maximum search and the nearest neighbor search.

Operations of this class are performed most efficiently in the appropriate special-purpose homogeneous processors, The ∇ -processor can be used as the device that performs a sufficiently broad set of reductive logic operations.

5. Data Structure Transformations.

The arguments and results are arrays of binary numbers (or bit strings) that have in the general case different dimensions and ranks. Processing consists in changing the structure of the array-argument or sequence of its components.

Operations of this class are efficiently performed by special-purpose devices for transforming data structures. In particular, a number of important operations of this class can be implemented by using the λ -processor.

Of course, the above classification is conventional. On the one hand, the high-volume statements could be combined into larger classes, making use of the fact that, for example, in an associative processor in principle, not only component-by-component, but also reductive operations are executable (although with less efficiency). On the other hand, perhaps it is advisable to separate out some of the statements included in one class (especially in the broad first class) and find for them a more efficient, narrowly specialized hardware implementation.

Nevertheless, it may be assumed that the number of functional modules sufficient for a considerable increase in efficiency of a system for high-volume data processing is not large: four to six.

That set of special-purpose processors in the aggregate can cover a broad assortment of high-volume statements, while achieving sufficient universality of hardware support. Required for efficient system operation, of course, is a good organization of distribution of tasks, loading of arrays-arguments and exchange of intermediate results.

34. Possibilities of Practical Implementation of Special-Purpose Homogeneous Processors

There are now concrete possibilities for the transition to practical manufacture and use of special-purpose homogeneous processors.

The main prerequisite for this is the possibility of constructing large, inexpensive circuits.

This circumstance, associated with progress in the technology of manufacturing integrated microcircuits, has led, as is well known, to the rapid development and distribution of microprocessors [77]. Series produced microprocessors make it possible to implement microcomputers, the architecture of which differs little from the traditional. Such machines, just as other general-purpose computers, possess limited "capabilities."

On the other hand, as the degree of integration increases, the difficulties grow in designing and testing the irregular circuits of traditional architecture, as well as in connecting the individual modules with a limited number of leads.

One way to surmount these difficulties is to develop special-purpose, funtional-oriented microprocessor LSI circuits which especially possess a regular structure. Such devices include the now mastered programmable logic arrays (PLM), special-purpose microprocessors for multiplication, fot executing fast Fourier transforms, etc. Some experts believe the main application of very large integrated micro-circuits will be special-purpose functional devices [9].

Two-dimensional homogeneous logic circuits are a convenient object for implementation with LSI circuits. Therefore it can be assumed that based on the structures of special-purpose homogeneous processors such as described in this book, it is expedient to develop some special-purpose LSI circuits that supplement the standard series of microprocessors and are compatible with them.

An important factor determining the selection of a specific structure for practical implementation of an LSI circuit is the flexibility of the circuit and the possibility of sufficiently broad application of a type module. From this viewpoint, special-purpose homogeneous processors have definite advantages.

With a rational design, a special-purpose homogeneous processor may execute not only that operation for which its logic network is properly oriented, but also a number of other operations. Let us note that the υ_{min}-processor executes with identical efficiency the basic operations PVM, PNB, PVB, PBM and PS [search for all lower, maximum search, search for all greater, search for nearest neighbor lower and coincidence search]; the λ-processor performs the basic operations of compression, expansion and others.

There are various ways to expand the functional capabilities of special-purpose homogeneous processors:

combining in one structure the functions of several different processors;

using a sequence of elementary (basic) operations of the processor to execute complex procedures. This is how, for example, the sorting algorithm in the υ-processor or the associative computations in the π-processor are implemented;

using some storage elements and (or) external leads for adjustment of the structure. The adjustment of a-matrices for execution of the different functions to transform data structures can serve as an example.

To illustrate the flexibility of special-purpose homogeneous processors, let us list some possibilities of a circuit with the simplest cells--the a-matrix.

By construction, the a_{max}-processor is designed to perform the basic operation of maximum search. It is well known that by microprogramming in the a-processor, various information retrieval algorithms can be implemented. By adjustment of the cells, the switching function can be implemented.

The method of modeling of associative memory in an a -structure is suggested in [78]. In principle, this makes it possible to organize various high-volume computations by the method of sequential transformation of states.

The possibility of implementing random finite automata in an a -structure is shown in [23].

A cell of associative memory built on the base of an a -structure has four steady states, and thanks to this the a -structure can efficiently be used as functional memory. P. Gardner, in [79], discusses tabular methods of computation of functions in homogeneous structures, and the presence of redundant states in associative memory cells is used for minimization. As shown in [80], the a -structure makes it possible to apply all the minimization methods suggested by Gardner and in addition, achieve a number of additional advantages.

An essential factor influencing the realization of some structure with LSI circuits is the limitation on the number of external connections. This requires serious study of the design of special-purpose homogeneous processors with regard to specific features of the technology applied. In doing so, it is necessary to make use of all known means to reduce the lead/element ratio (incorporation within the module of circuits for control, encoding, decoding, multiplex mode of exchange, etc.).

With some types of special-purpose homogeneous processors, the problem is simplified because a relatively small set of standard vectors is sufficient for control. These vectors can be generated by special circuits included within the design modules and controlled from without by feeding economical codes ("numbers") of the vectors needed.

Solving the problem of leads is substantially facilitated in going from static homogeneous structures to dynamic (chapter 7). This makes it possible to consider that dynamic parallel processors are now closest to practical use.

BIBLIOGRAPHY

1. Kantorovich, L. V., "Paths of Development of Computer Facilities for Solving Large-Scale Problems of Optimal Planning and Control," in the book "Optimizatsiya (sbornik trudov)" [Optimization (Collection of Works)], Novosibirsk, Institute of Mathematics, Siberian Branch of the USSR Academy of Sciences, 1972, issue 6(23), pp 5-7.
2. "Multiprocessor Systems and Parallel Computing," edited by F. G. Enslow, translated from English, Moscow, Mir, 1976, 384 pages.
3. Kantorovich, L. V., "A Mathematical Notation Suitable When Calculations Are Made on a Machine," DOKLADY AN SSSR, 1957, Vol 113, No 4, pp 738-741.
4. Kantorovich, L. V., "Prospects of Efforts to Automate Programming Based on a Large-Block System," TRUDY MAT. IN-TA IMENI V. A. STEKLOVA, 1968, Vol 96, pp 5-15.

5. Glushkov, V. M.; Barabanov, A. A.; Kalinichenko, L. A. et al., "Vychislitel'nyye machiny s razvitymi sistemami interpretatsii" [Computers with Developed Systems of Interpretation], Kiev, Naukova dumka, 1970, 260 pages.

6. Zadykhaylo, I. B.; Kamynin, S. S. and Lyubimskiy, E. Z., "Questions of Design of Computers from Blocks of Enhanced Qualification," in the book "Sistemnoye i teoreticheskoye programmirovaniye (sbornik trudov)" [Systems and Theoretical Programming (Collection of Works)], Novosibirsk, Computer Center, Siberian Branch, USSR Academy of Sciences, 1972, pp 126-135.

7. Sima, M. "16-Bit Instrument That Implements Microprocessor and Minicomputer Functions," ELEKTRONIKA, Russian translation, Vol 51, No 26, 1978, pp 24-34.

8. Turuta, Ye. N., "Multimicroprocessor Systems," ZARUBEZHNAYA RADIOELEKTRONIKA, No 3, 1979, pp 3-27.

9. Keypis, R. P., "Problems of Design and Production of Very Large-Scale Integrated Circuits," ELEKTRONIKA, Russian translation, Vol 51, No 24, 1978, pp 29-51.

10. Faddeyeva, V. N. and Faddeyev, D. K., "Parallel Computations in Linear Algebra," KIBERNETIKA, No 6, 1977, pp 28-40.

11. Flynn, M. G., "High-Speed Computing Systems," TRANS. IEEE, Russian translation, Vol 54, No 12, 1966, pp 311-320.

12. Yevreinov, E. V., "Microstructure of Elementary Machines of Computer System," in the book "Vychislitel'nyye sistemy (sbornik trudov)" [Computer Systems (Collection of Works)], Novosibirsk, Institute of Mathematics, Siberian Branch, USSR Academy of Sciences, issue 4, 1962, pp 5-28.

13. Hennie, F. C., "Iterative Arrays of Logical Circuits," New York-London, MIT Press-Wiley, 1961, 241 pages.

14. Thurber, K. J. and Wald, L.D., "Associative and Parallel Processors," ACM COMPUTING SURVEYS, Vol 7, No 4, 1975, pp 215-255.

15. Hennie, F. C., "Finite-State Models for Logical Machines," New York-London-Sydney, Wiley, 1968, 466 pages.

16. Varshavskiy, V. I.; Marakhovskiy, V. B. and Peschanskiy, V. A. et al., "Odnorodnyye struktury/ (Analiz. Sintez. Povedeniye)/" [Homogeneous Structures: Analysis, Synthesis, Behavior], Moscow, Energiya, 1973, 152 pages.

17. Majithia, J. C. and Kitai, R., "An Iterative Array for Multiplication of Signed Binary Numbers," IEEE TRANS., Vol C-20, No 2, 1971, pp 214-216.

18. Toma, C. I., "Cellular Logic Array for High-Speed Signed Binary Multiplication," IEEE TRANS., Vol C-24, No 9, 1975, pp 932-935.

19. Guild, H. H., "Some Cellular Logic Arrays for Non-restoring Binary Division," RADIO ELECTRON. ENG., Vol 39, No 6, 1970, pp 345-348.

20. Cappa, M. and Hamacher, V. C., "An Augmented Iterative Array for High-Speed Binary Division," IEEE TRANS., Vol C-22, No 2, 1973, pp 172-175.

21. Shauman, A. M., "Matrix Extraction of Square Root," in the book "Vychislitel'naya tekhnika" [Computer Technology], Leningrad, Leningrad State University, issue 6, 1977, pp 105-111.

22. Fet, Ya. I., Patent 424141 (USSR), "Ustroystvo dlya sortirovki informatsii" [Information Sorting Device], published in B. I., No 14, 1974.

23. Fet., Ya. I., "Massovaya obrabotka informatsii v spetsializirovannykh odnorodnykh protsessorakh" [High-Volume Information Processing in Special-Purpose Homogeneous Processors], Novosibirsk, Nauka, 1976, 200 pages.

24. Blokh, A. Sh. and Lades, V. I., "Synthesis of Single-Cycle Circuits, Behavior of Which Is Described by Linear Inequalities," IZV. AN BSSR. SER. FIZ-TEKHN. NAUK, No 3, 1966, pp 93-99.

25. Berkovich, S. Ya. and Kochin, Yu. Ya., "Nearest Neighbor Search," AVTOMATIKA I TELEMEKHANIKA, No 2, 1975, pp 176-178.

26. Fet, Ya. I., Patent 634372 (USSR), "Element assotsiativnoy matritsy pamyati" [Associative Memory Matrix Element], published in B. I., No 43, 1978.

27. Kratko, M. I., "Regular and Stable Iterative Networks," in the book "Problemy kibernetiki" [Problems of Cybernetics], Moscow, Nauka, issue 19, 1967, pp 95-106.

28. Glushkov, V. M.; Gladun, V. P.; Lozinskiy, L. S. et al., "Obrabotka informatsionnykh massivov v abtomatizirovannykh sistemakh upravleniya" [Information File Processing in Automated Control Systems], Kiev, Naukova dumka, 1973, 183 p.

29. Papernov, A. A. and Podymov, V. Ya., "Metody uporyadocheniya informatsii v tsifrovykh sistemakh" [Information Sorting Methods in Digital Systems], Moscow, Nauka, 1973, 384 pages.

30. Flores, I., "Analysis of Internal Computer Sorting," JOURNAL OF THE ACM, Vol 8, No 1, 196], pp 41-80.

31. Martin, W. A., "Sorting," ACM COMPUTING SIRVEYS, Vol 3, No 4, 1971, pp 147-174.

32. Krayzmer, L. P.; Borodayev, D. A.; Gutenmakher, L. I. et al., "Assotsiativnyye zapominayushchiye ustroystva" [Associative Memory], Leningrad, Energiya, 1967, 184 pages.

33. Kautz, W. H., "Cellular Logic-in-Memory Arrays," IEEE TRANS., Vol C-18, No 8, 1969, pp 719-727.

34. Bratal'skiy, Ye. A. and Krupskiy, A. A., "File Sorting Method Using Associative Device," VOPROSY RADIOELEKTRONIKI. SER. EVT, issue 7, 1973, pp 90-93.

35. Batcher, K. E., "Sorting Networks and Their Applications," in AFIPS CONFER. PROC., 1968 SJCC, Vol 32, pp 307-314.

36. Fuller, R. H. and Estrin, G., "Some Applications for Content-Addressable Memories," in AFIPS CONFER. PROC., 1963 FJCC, Vol 24, pp 495-508.

37. Shooman, W., "Parallel Computing with Vertical Data," in AFIPS CONFER. PROC., 1960 EJCC, Vol 18, pp 111-115.

38. Shooman, W., Patent 3277449 (USA), "Orthogonal Computer."

39. Shooman, W., "Orthogonal Processing," in "Parallel Processors, Systems, Technologies and Applications," Spartan Books, 1970, pp 297-308.

40. Crane, B. A. and Githens, J. A., "Bulk Processing in Distributed Logic Memory," IEEE TRANS., Vol EC-14, No 2, 1965, pp 186-196.

41. Lipovski, G. J., "The Architecture of a Large Associative Processor," in AFIPS CONFER. PROC., 1970 SJCC, Vol 36, pp 385-396.

42. Popova, G. M. and Prangishvili, I. V., "Associative Parallel Processor for for Batch Data Processing," AVTOMATIKA I TELEMEKHANIKA, No 1, 1972, pp 171-184.

43. Aleyeva, V. N. and Fet, Ya. I., "Associative Processor for Large-Block Processing of Information," in the book "Optimizatsiya (sbornik trudov)" [Optimization (Collection of Works)], Novosibirsk, Institute of Mathematics, Siberian Branch, USSR Academy of Sciences, issue 15(32), 1974, pp 154-177.

44. Prangishvili, I. V.; Popova, G. M.; Smorodinova, O. G.; and Chudin, A. A., "Odnorodnyye mikroelektronnyye assotsiativnyye protsessory" [Homogeneous Microelectronic Associative Processors], Moscow, Sovetskoye Radio, 1973, 280 pages.

45. Fet, Ya. I., Patent 746728 (USSR), "Zapominayushchiy modul' dlya matrichnykh blokov pamyati" [Storage Module for Matrix Blocks of Memory], published in B. I., No 25, 1980.

46. Iverson, K. E., "A Programming Language," New York-London, Wiley, 286 pages.

47. Pakin, S., "APL/360 Reference Manual," Chicago, Science Research Associates, 1968, 160 pages.

48. Abrams, P. S., "An APL Machine," Stanford Univ., AD-706741, 1970, 213 pages.

49. Thurber, K. J. and Myrna, J. W., "System Design of a Cellular APL Machine," IEEE TRANS., Vol C-19, No 4, 1970, pp 291-303.

50. Harrison, M. J. and Harrison, W. H., "The Implementation of APL on Associative Processor," in "Parallel Processing (Proc. of the Sagamore Comp. Confer., 1974)," Berlin-Heidelberg-New York, Springer, 1975, pp 75-96.

51. Higbie, L. C., "The OMEN Computers: Associative Array Processors," in COMPCON 72, 6-th Annual IEEE Comp. So. Int. Confer., San Francisco, 1972, pp 287-290.

52. Slotnick, D. L., "Logic-per-Track Devices," in "Advances in Computers," New York, Academic Press, Vol 10, 1970, pp 291-296.

53. Parker, J. L., "A Logic per Track Retrieval System," in "Proc. of IFIP Congress 71," Amsterdam, North-Holland Publ. Comp., 1971, Vol 1, pp 711-716.

54. Coulouris, G. F.; Ewans, J. M. and Mitchell, R. W., "Towards Content-Addressing in Data Bases," COMPUT. JOURN., Vol 15, No 2, 1972, pp 95-98.

55. Finnila, C. A. and Love, H. H., "The Associative Linear Array Processor," IEEE TRANS., Vol C-26, No 2, 1977, pp 112-125.

56. Anderson, G. A. and Kain, R. Y., "A Content-Addressed Memory Designed for Data-Base Applications," in "Proc. 1976 Int. Confer. on Parallel Processing," New York, IEEE, 1976, pp 191-195.

57. Banerjee, J.; Hsiao, D. K. and Kannan, K., "DBC--A Database Computer for Very Large Databases," IEEE TRANS., Vol C-28, No 6, 1979, pp 414-429.

58. Su, S. Y. W.; Nguyen, L. H.; Emam, A and Lipovski, G. J., "The Architectural Features and Implementation techniques of the Multicell CASSM," IEEE TRANS., Vol C-28, No 6, 1979, pp 430-445.

59. Thurber, K. J., "Interconnection Networks--A Survey and Assessment," in AFIPS Nat. Comp. Confer., Vol 43, 1974, pp 909-919.

60. Kalyayev, A. V., "Odnorodnyye kommutatsionnyye registrovyye struktury" [Homogeneous Switch Register Structures], Moscow, Sovetskoye radio, 1978, 336 p.

61. Feng, T.-Y., "Data Manipulating Functions in Parallel Processors and Their Implementations," IEEE TRANS., Vol C-23, No 3, 1974, pp 309-318.

62. Stone, H. S., "Parallel Processing with the Perfect Shuffle," IEEE TRANS., Vol C-20, No 2, 1971, pp 153-161.

63. Benes, V. E., "Optimal Rearrangeable Multistage Connecting Networks," BSTJ, Vol 43, No 4, 1964, pp 1641-1656.

64. Opferman, D. C. and Tsao-Wu, N. T., "On a Class of Rearrangeable Switching Networks," BSTJ, Vol 50, No 5, 1971, pp 1579-1618.

65. Kautz, W. H.; Levitt, K. N. and Waksman, A., "Cellular Interconnection Arrays," IEEE TRANS., Vol C-17, No 5, 1968, pp 443-451.

66. Waksman, A., "A Permutation Network," JOURN. ACM, Vol 15, No 1, 1968, pp 159-163.

67. Pease, M. C., "An Adaptation of the Fast Fourier Transform to Parallel Processing," JOURN. ACM, Vol 15, No 4, 1968, pp 252-264.

68. Lang, T. and Stone, H. S., "A Shuffle-Exchange Network with Simplified Control," IEEE TRANS., Vol C-25, No 1, 1976, pp 55-65.

69. Aleyeva, V. N. and Fet, Ya. I., Patent 590747 (USSR), "Dvumernaya odnorodnaya struktura dlya analiza logicheskikh vektorov" [Two-Dimensional Homogeneous Structure for Analysis of Logic Vectors], published in B. I., No 4, 1978, p 188.

70. Aleyeva, V. N., "Solving Problems of Linear Algebra with Associative Processor," in the book "Optimizatsiya (sbornik trudov)" [Optimization (Collection of Works)], Novosibirsk, Institute of Mathematics, Siberian Branch, USSR Academy of Sceinces, issue 18(35), 1976, pp 146-159.

71. Okhotsimskiy, D. Ye. and Platonov, A. K., "Algorithms to Control Walking Apparatus Capable of Surmounting Obstacles," IZV. AN SSSR. TEKHNICHESKAYA KIBERNETIKA, No 5, 1973, pp 3-10.

72. Barsamian, H., "Firmware Sort-Processor with LSI Components," in AFIPS Confer. Proc., 1970 SJCC, Vol 36, pp 183-190.

73. De Fiore, C. R., "Fast Sorting," DATAMATION, Vol 16, No 8, 1970, pp 47-51.

74. De Fiore, C. R. and Berra, P. B., "A Quantitative Analysis of the Utilization of Associative Memories in Data Management," IEEE TRANS., Vol C-23, No 2, 1974, pp 121-133.

75. Fet, Ya. I., "Hardware Support of High-Volume Computations," in the book "Optimizatsiya (sbornik trudov)" [Optimization (Collection of Works)], Novosibirsk, Institute of Mathematics, Siberian Branch, USSR Academy of Sciences, issue 22(39), 1978, pp 115-126.

76. Fet. Ya. I., "Implementation of Group Operations on Rotor Conveyor Processor," in the book "Optimizatsiya (sbornik trudov)" [Optimization (Collection of Works)], Novosibirsk, Institute of Mathematics, Siberian Branch, USSR Academy of Sciences, issue 6(23), 1972, pp 104-115.

77. Prangishvili, I. V., "Mikroprotsessory i mikro-EVM" [Microprocessors and Microcomputers], Moscow, Energiya, 1979, 232 pages.

78. Fet, Ya. I., "Functional Possibilities of Simple Computing Media," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, Riga, No 3, 1974, pp 48-54.

79. Gardner, P. L., "Functional Memory and Its Microprogramming Implications," IEEE TRANS., Vol C-20, No 7, 1971, pp 765-775.

80. Fet. Ya. I., "Special-Purpose Homogeneous Structures as Programmable Logic Arrays," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, Riga, No 6, 1979, pp 67-72.

8545
CSO: 1863/251

END